МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ У ТЕХНІЧНИХ СИСТЕМАХ

Проектування мікропроцесорних систем:
Кредитний модуль «Проектування мікропроцесорних систем на базі мікроконтролерів сімейства MCS-51»
Програмування мікропроцесорних систем на базі мікроконтролерів сімейства MCS–51 : Навчальний посібник

для студентів напряму підготовки 6.050201 «Системна інженерія»

Київ НТУУ “КПІ” 2016

Проектування мікропроцесорних систем: Кредитний модуль «Проектування мікропроцесорних систем на базі мікроконтролерів сімейства MCS-51»: Програмування мікроконтролерів сімейства MCS–51: Навчальний посібник для студентів напряму підготовки 6.050201 «Системна інженерія» кафедри Автоматики та управління у технічних системах / Автор: А.О. Новацький– К: НТУУ „КПІ”, 2016–156с.

Відповідальний за випуск: к. т. н., доц. Л.Ю. Юрчук

Рецензенти: доктор фіз. мат. наук, професор А.Ю.Дорошенко,

к. т. н., доц. О.А.Чемерис

Надано гриф «Затверджено Вченою радою НТУУ «КПІ» як навчальний посібник для студентів, які навчаються за спеціальністю «Автоматизація та комп`ютерно–інтегровані технології»

(Протокол №1 від 18 січня 2016 року)

Навчальний посібник охоплює теоретичний матеріал та практичні завдання, які необхідні для вивчення базової частини дисципліни «Проектування мікропроцесорних систем». Навчальний посібник складається із чотирьох розділів: склад та основні характеристики мікроконтролерів сімейства МК–51, структура типового мікроконтролера МК–51, програмна модель мікроконтролера та характеристика команд. В роботі розглядаються основні архітектурні особливості мікроконтролерів сім`ї MCS–51 на апаратному рівні, після чого основна увага приділяється особливостям програмування цих мікроконтролерів. Робота може бути корисною студентам відповідних спеціальностей при вивченні дисциплін, які пов’язані із проектуванням та використанням мікропроцесорних систем, а також при виконанні бакалаврських робіт, курсових та дипломних проектів, в яких використовуються мікропроцесорні пристрої. Останнє було враховано при оформленні роботи, яку виконано згідно вимог до конструкторської документації.

ЗМІСТ

ВСТУП

MCS–51 – сімейство мікроконтролерів, які розроблено фірмою Intel у 1980 році для використання у вбудованих системах. Архітектура сімейства MCS–51 була настільки вдалою, що стала стандартом на світовому ринку 8–розрядних мікроконтролерів.

MCS–51 прийшли на заміну випущеним у 1976 році MCS–48 і на відміну від останніх мали зменшений час виконання команд (в 2,5 – 10 раз в залежності від умов експлуатації), збільшений обсяг вбудованої пам'яті, додаткові пристрої периферії, додаткові команди для програмування. За рахунок даних покращень, мікроконтролери стали зручнішими в програмуванні, дешевші в експлуатації.

Конструктивно, MCS–51 є однокристальним мікроконтролером гарвардської архітектури, який виконано за n–МОН або КМОН–технологіями. Він містить у собі 8–бітний мікропроцесор і8051 з підтримкою булевих операцій над окремими бітами, до 4096 байт вбудованої пам'яті програм (доступної тільки для читання), до 256 байт вбудованої пам'яті даних (доступної для читання і запису), підтримка адресного простору у 64 Кб для пам'яті програм і 64Кб для пам'яті даних, два 16–бітні таймери/лічильники, двосторонній УАПП, генератор тактової частоти. Для підготовки програмного забезпечення мікроконтролерів MCS–51 використовуються в основному мови "ASM–51" і "C", для яких існують ряд компіляторів, бібліотек, стандартних підпрограм і програмних емуляторів, вироблених різними закордонними та вітчизняними фірмами.

В даний час серед всіх 8–розрядних мікроконтролерів – сімейство MCS–51 є безсумнівним чемпіоном по кількості різновидів і кількості компаній, що випускають його модифікації. Воно отримало свою назву від першого представника цього сімейства – мікроконтролера 8051,

випущеного в 1980 році. Вдалий набір периферійних пристроїв, можливість гнучкого вибору зовнішньої або внутрішньої програмної пам'яті і прийнятна ціна забезпечили цьому мікроконтролеру успіх на ринку.

Важливу роль у досягненні такої високої популярності сімейства 8051 зіграла відкрита політика фірми Intel, родоначальниці архітектури, яка спрямована на широке поширення ліцензій на ядро 8051 серед великої кількості провідних напівпровідникових компаній світу. Фірма Intel досить давно не займається випуском 8–розрядних мікропроцесорів. Велика кількість виробників придбали ліцензію на випуск мікроконтролерів з ядром MCS–51: Philips, Siemens, Dallas, Atmel, Winbond та інші. Всі компанії прагнуть одночасно зберегти сумісність зі стандартною архітектурою MCS51 і привнести в неї які–небудь покращення, дозволяючи продукції виділятися на фоні продуктів конкуруючих фірм.

В результаті на сьогоднішній день існує більше 200 модифікацій мікроконтролерів сімейства 8051. Ці модифікації містять у собі кристали з найширшим спектром периферії: від простих 20–контактних пристроїв з одним таймером і 1К програмної пам'яті до найскладніших 100–контактних кристалів з 10–розрядними АЦП, масивами таймерів–лічильників, апаратними 16–розрядними множниками і 64К програмної пам'яті на кристалі.

Мікроконтролери фірми Atmel – найбільш популярні серед розробників з країн СНД. Цьому сприяє їх невисока ціна і наявність в гамі продукції великої кількості мікроконтролерів з флеш–пам’яттю програм, що особливо важливо для дрібносерійного виробництва. Поряд з повними стандартними версіями контролерів, фірма Atmel пропонує більш дешеві кристали з меншою кількістю виводів (20 проти 40 стандартних).

Однією із найавторитетніших компаній на ринку мікроконтролерів є Siemens. В результаті реорганізації підрозділ по випуску напівпровідників отримало свою власну назву – Infineon. Фірма Infineon випускає, разом з іншими цікавими мікросхемами, мікроконтролери сімейства C500. Архітектура C500 базується на MCS–52, але має більш розвинуту периферію.

Велику номенклатуру мікроконтролерів з ядром MCS–51 випускає фірма Philips. Продукція фірми орієнтована, в першу чергу, на ринок побутової електроніки. Більшість кристалів мають інтерфейс I2C. I2C – це послідовна шина, яку запропоновано фірмою Philips в якості стандарту обміну в недорогих низькошвидкісних системах.

Analog Devices вийшли на ринок 8–розрядних мікроконтролерів, запропонувавши сімейство MicroConverter. У цих виробах об'єднано на одному кристалі найважливіші компоненти систем збору і обробки інформації – процесорне ядро MCS–52 і блок введення/виведення аналогової інформації, що включає багатоканальні ЦАП і АЦП.

Особливістю мікроконтролерів фірми Dallas Semiconductor є змінена часова сітка. Машинний цикл в даних мікропроцесорах займає 4 такти замість 12 у класичного 80C51. При цьому збережено повна сумісність зі стандартною архітектурою на рівні інструкцій. Мікроконтролери Dallas знаходять застосування в задачах, які потребують підвищеної швидкості обробки.

Фірма Silicon Laboratories динамічно розвиває перелік мікроконтролерів. Фірмі вдалося створити більше 55 типів мікроконтролерів, що відрізняються продуктивністю, об'ємом Flash–пам'яті програм, вбудованої оперативної пам'яті, характеристиками аналого–цифрових вузлів, типом корпусу та іншими параметрами. Перевагою мікроконтролерів цього сімейства є те, що один машинний

цикл має один такт. Це значно підвищує швидкодію мікроконтролера. При цьому збережено повна сумісність зі стандартною архітектурою на рівні інструкцій.

Всі мікроконтролери з сімейства MCS–51 мають спільну систему команд. Наявність вбудованого додаткового периферійного обладнання впливає тільки на кількість регістрів спеціального призначення.

Поява 16– і 32–розрядних мікроконтролерів та цифрових сигнальних процесорів, які значно переважають 8–розрядні за продуктивністю, не витіснило їх. Більш того, за кількістю модифікацій 8–розрядні мікроконтролери значно перевершують всі інші групи. Головна причина криється в тому, що основна область застосування 8–розрядних мікроконтролерів – пристрої інтелектуального керування промислової автоматики та побутової апаратури. Специфіка алгоритмів керування цих пристроїв не вимагає виконання розрахунків високої точності в жорстких умовах реального часу. Основна частина операцій керування полягає в перетворенні логічної інформації, і 8–розрядні мікроконтролери з успіхом реалізують ці завдання. За рахунок вдалої реалізації мікроконтролера, велика кількість наявних на ринку мікроконтролерів має i8051 сумісні процесори, тому ядро MCS–51 є одним з перших кроків до вивчення сучасних мікропроцесорів у програмах курсів вищих навчальних закладах. Мікроконтролери сімейств MCS–51 фірми Intel і HC05 фірми Motorola рекомендовані типовою програмою для вивчення студентами радіотехнічних спеціальностей.

Незважаючи на солідний вік сімейства і появу на світовому ринку однокристальних мікроконтролерів більшої продуктивності і кращої архітектури, контролери MCS–51 ще достатньо довго будуть використовуватися в простих вбудованих системах керування та автоматики.

СПИСОК СКОРОЧЕНЬ

АЛП – арифметико–логічний пристрій;

АЦП – аналого–цифровий перетворювач;

БО – безпосередній операнд;

ГТІ – генератор тактових імпульсів;

ДК – двійковий (машинний) код;

ДНЛ – диференціальна нелінійність;

ЕОМ – електронно–обчислювальна машина;

ЗПД – зовнішня пам'ять даних;

ЗПП – зовнішня пам'ять програм;

КМОН – комплементарний метал–оксид–напівпровідник;

КОП – код операції команди;

КЦ – командний цикл;

ЛК – лічильник команд;

МА – мова асемблера;

МЗР – молодший значущий розряд;

МК – мікроконтролер;

МПС – мікропроцесорна система;

НОЗП – над оперативний запам’ятовуючий пристрій;

ОЗП – оперативний запам’ятовуючий пристрій;

ПВВ – пристрої введення/виведення;

ПЗП – постійний запам’ятовуючий пристрій;

ПЛМ – програмована логічна матриця;

ПП – пам’ять програм;

ППЗП – перепрограмовуваний постійний запам'ятовуючий пристрій;

ППР – периферійний пристрій;

РА (RAR) – регістр адреси;

РЗП – регістр загального призначення;

РК (IR) – регістр команд;

РКП (PCON) – регістр керування потужністю споживання енергії від джерела живлення;

РКПП (SCON) – регістр керування приймачем–передавачем послідовного порту;

РКСТ – регістр керування–статусу таймерів/лічильників

РМП – регістр масок переривань;

РП (IP) – регістр пріоритетів переривань;

РПД – резидентна пам'ять даних;

РПП – резидентна пам'ять програм;

РПС (SP) – регістр–покажчик стеку;

РРТЛ – регістр режимів таймерів/лічильників;

РСП (PSW) – регістр стану програми (прапорців);

РУПП -– регістр керування приймачем–передавачем;

РШД – резидентна шина даних;

СЗР – старший значущий розряд;

Т/Л – таймер/лічильник;

УАПП – універсальний асинхронний програмований приймач–передавач

УСАПП – універсальний синхронно/асинхронний приймач–передавач

ФАПЧ – фазове автопідстроювання частоти;

ЦАП – цифро–аналоговий перетворювач;

ША – шина адреси;

ШД – шина даних;

ШІМ – широтно–імпульсна модуляція;

BCD (binary–coded decimal) – двійко́во–десятко́вий код

DPTR (РГПД) – регістр–покажчик даних, що складається з 2–х частин: молодшої – DPL і старшої – DPH;

EEPROM–пам’ять (Electrically Erasable Programmable Read–Only Memory) ‑ енергонезалежна пам'ять, що може бути електрично стерта та перепрограмована;

IEEE (Institute of Electrical and Electronics Engineers) – інститут інженерів з електротехніки та електроніки;

ISP (In System Programarable) – режим внутрішньо системного програмування;

I2C (InterIC, або IIC) – двонаправлена двопровідна шина для так званого застосування між мікросхемами;

JTAG (Joint Test Action Group) ‑ інтерфейс, який призначено для підключення складних цифрових мікросхем або пристроїв рівня друкованої плати до стандартної апаратури тестування і налагодження;

PC – програмний лічильник;

SBUF (буфер ПРМ і буфер ПД) – буфери приймача і передавача послідовного порту;

SCL (Serial CLock) – послідовна лінія тактування;

SFR (Special Function Register) – регістр спеціальних функцій;

SPI (Serial Peripheral Interface) ‑ послідовний периферійний інтерфейс;

TWI (Two Wire (Serial) Interface) – двопровідний інтерфейс;

UART (universal asynchronous receiver/transmitter) – універсальний асинхронний приймач/передавач;

USART (Universal Synchronous and Asynchronous Receiver and Transmitter) – універсальний синхронний/асинхронний приймач/передавач.

1 СКЛАД ТА ОСНОВНІ ХАРАКТЕРИСТИКИ МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА МК–51

1.1 Загальні відомості

Фірма Intel досить давно не займається випуском 8–розрядних мікропроцесорів. Велика кількість виробників придбали ліцензію на випуск мікроконтролерів з ядром MCS–51: Philips, Siemens, Dallas, Atmel, Winbond, Silicon Laboratories та інші. Всі компанії прагнуть одночасно зберегти сумісність зі стандартною архітектурою MCS–51 і привнести в неї будь–які покращення, дозволяючи продукції виділятися на фоні продуктів конкуруючих фірм.

1.2 Мікроконтролери фірми Atmel

Мікроконтролери фірми Atmel – мають велику популярність серед розробників з країн СНД. Цьому сприяє їх невисока ціна і наявність в гамі продукції великої кількості мікроконтролерів з флеш–пам’яттю програм, що особливо важливо для дрібносерійного виробництва. Поряд з повними стандартними версіями контролерів, фірма Atmel пропонує більш дешеві кристали з меншою кількістю виводів (20 проти 40 стандартних).

В таблиці 1.1 приведено характеристики деяких мікроконтролерів фірми Atmel з флеш–пам’яттю програм. Фірма випускає також одноразово програмовані версії кристалів, але вони не користуються такою популярністю.

1.3 Мікроконтролери Temic

Фірма Temic стала підрозділом Atmel із назвою Atmel Wireless. На момент об’єднання у цієї фірми був багаторічний досвід розробки та випуску мікроконтролерів з ядром MCS–52.

Таблиця 1.1 – Мікроконтролери фірми Atmel

Тип	Напруга живлення, В	Тактова частота, МГц	I/O	FLASH	EEPROM	SRAM	Інтерфейси	Аналогові інтерфейси	Таймери	Корпус
Мікроконтролери з Flash–пам'яттю програм										
AT89C2051	2.7–6.0	24	15	2K	–	128	UART	Comp	2x16–bit	PDIP20, SOIC20
AT89C4051	3.0–5.0	24	15	4K	–	128	UART	Comp	2x16–bit	PDIP20, SOIC20
AT89C51	5.0	24	32	4K	–	128	UART	–	2x16–bit	DIP40, PLCC44, PQFP44
AT89LV51	2.7	12	32	4K	–	128	UART	–	2x16–bit	DIP40, PLCC44, PQFP44
AT89C52	5.0	24	32	8K	–	256	UART	–	3x16–bit	DIP40, PLCC44, PQFP44
AT89LV52	2.7	12	32	8K	–	256	UART	–	3x16–bit	DIP40, PLCC44, PQFP44
AT89C55WD	5.0	33	32	20K	–	256	UART	–	3x16–bit	DIP40, PLCC44, PQFP44
AT89LV55	2.7	12	32	20K	–	256	UART	–	3x16–bit	DIP40, PLCC44, PQFP44
AT89C51RC	5.0	33	32	32K	–	512	UART	–	3x16–bit PCA	DIP40, PLCC44, PQFP44
Мікроконтролери з інтерфейсом USB, які програмуються внутрішньосхемно										
AT89C51SND1	2.7–3.3	20	44	64K	–	2304	UART, IDE, USB, SPI, I^2S, MP3 Decoder	10–bit ADC	2x16–bit WDT	TQFP80, PLCC84
AT89C51SND2	2.7–3.3	20	44	64K	–	2304	UART, IDE, USB, SPI, I^2S, MP3 Decoder	10–bit ADC, 2x20bit DAC	2x16–bit WDT	CTBGA100
AT89C5131	3.0–3.6	40	34 18	32K	4Kb	1280	UART, USB, SPI	–	3x16–bit WDT, PCA	PLCC52, VQFP64, MLF48, SO28

Продовження таблиці 1.1

AT89C5131A–L	3.0–3.6	48	34 18	32K	4Kb	1280	UART, USB, SPI, I^2C	–	3x16–bit WDT, PCA	PLCC52, VQFP64, MLF48, SO28
AT89C5130A–M	2.7–5.5	48	34	16K	–	1280 + DPRAM 1280	UART, USB, SPI	–	2x16–bit WDT	PLCC52, VQFP64, QFN32
AT89C5131A–M	3.0–3.6	40	34 18	32K	256	1024	USB, SPI	–	2x16–bit WDT	LCC52, VQFP64, MLF48, SO28
AT89C5132	2.7–3.3	40	44 38	64K	–	2304	UART, USB, SPI, I^2S	2ch 10bit	3x16–bit WDT	TQFP80, TQFP64
AT85C5122	3.6–5.5	16	46 13	32K CRAM		768	UART, USB, SPI, Smart Card	–	2x16–bit WDTA	VQFP64, PLCC28
AT89C5122	3.6–5.5	16	46 13	32K		768	UART, USB, SPI, Smart Card	–	2x16–bit WDTA	VQFP64, PLCC28
Мікроконтролери, які програмуються внутрішньосхемно										
T85C5121	3.6–5.5	16	14 30	16K CRAM		768	UART, SPI, Smart Card	–	2x16–bit WDTA	SSOP24, PLCC52
T89C5121	3.6–5.5	16	14 30	16K		768	UART, SPI, Smart Card	–	2x16–bit WDTA	SSOP24, PLCC52
AT89S51	4.0–5.5	33	32	4K	–	128	UART	–	2x16–bit WDT	DIP40, PLCC44, PQFP44
AT89LS51	2.7–4.0	16	32	4K	–	128	UART	–	2x16–bit WDT	DIP40, PLCC44, PQFP44
AT89S2051	2.7–5.5	24	15	2K	–	256	UART	–	2x16–bit	PDIP20, SO20
AT89S4051	2.7–5.5	24	15	4K	–	256	UART	–	2x16–bit	PDIP20, SO20
AT89S52	4.0–5.5	33	32	8K	–	256	UART	–	3x16–bit WDT	DIP40, PLCC44, PQFP44
AT89LS52	2.7–4.0	33	32	8K	–	256	UART	–	3x16–bit WDT	PDIP40, PDIP42, PLCC44, TQFP44
AT89S8252	4.0–6.0	24	32	8K	2K	256	UART, SPI	–	3x16–bit WDT	DIP40, PLCC44, PQFP44
AT89S8253	2.7–5.5	24	32	12K	2K	256	UART, SPI	–	3x16–bit WDT	DIP40, PLCC44, PQFP44, PDIP42

Продовження таблиці 1.1

AT89LS8252	2.7–6.0	12	32	8K	2K	256	UART, SPI	–	3x16–bit WDT	DIP40, PLCC44, PQFP44
AT89S53	4.0–6.0	24	32	12K	–	256	UART, SPI	–	3x16–bit WDT	DIP40, PLCC44, PQFP44
AT89LS53	2.7–6.0	12	32	12K	–	256	UART, SPI	–	3x16–bit WDT	DIP40, PLCC44, PQFP44
T89C51RB2M	4.5–5.5	40	32	16K	–	1280	UART, SPI	–	3x16–bit PCA, WDT	DIP40, PLCC44, VQFP44
T89C51RB2L	2.7–3.3	40	32	16K	–	1280	UART, SPI	–	3x16–bit PCA, WDT	DIP40, PLCC44, VQFP44
T89C51RC2	4.5–5.5	40	32	32K	–	1280	UART, SPI	–	3x16–bit PCA, WDT	DIP40, PLCC44, VQFP44
T89C5115	4.5–5.5	40	20	16K	2K	512	UART	10–bit ADC	3x16–bit PCA, WDT	SOIC28, PLCC28, VQFP32
T89C51IC2	2.7–3.6 4.5–5.5	40	32	32K	–	1280	UART, SPI, I^2C	–	3x16–bit PCA, WDT	PLCC44, VQFP44
AT89C51ID2	2.7–3.6 3.0–5.5	40	32 48	64K	2K	2K	UART, SPI, I^2C	–	3x16–bit PCA WDT	PLCC44, VQFP44 PLCC68, VQFP64
T89C51AC2	4.5–5.5	40	34	32K	2K	1280	UART	10–bit ADC	3x16–bit PCA, WDT	VQFP44, PLCC44, CA–BGA64
AT89C51AC3	3.0–5.5	60	36	64K	2K	2304	UART, SPI	10–bit ADC	3x16–bit PCA, WDT	VQFP44, PLCC44, VQFP64, PLCC52
T89C51RD2	3.0–5.5 4.5–5.5	40	32 48	64K	2K	1280	UART	–	3x16–bit PCA, WDT	PDIL40, PLCC44, VQFP44, PLCC68, VQFP64
AT89C51RD2	2.7–5.5	40	32 48	64K	–	1280	UART	–	3x16–bit PCA, WDT	PLCC44, VQFP44
AT89C51ED2	2.7–5.5	40	32 48	64K	2K	2048	UART	–	3x16–bit PCA, WDT	PDIL40, PLCC44, VQFP44, PLCC68, VQFP64

Продовження таблиці 1.1

AT89C51IC2	2.7–5.5	40	34 48	32K	–	1280	UART, TWI	–	3x16–bit PCA, WDT	LQFP44, PLCC44
AT89C51RB2	2.7–5.5	40	32 48	16K	–	1280	UART	–	3x16–bit PCA, WDT	PDIP40, LQFP44, PLCC44
T89C51CC01	4.5–5.5	40	34	32K	2K	1280	UART, CAN	10–bit ADC	3x16–bit PCA, WDT	TQFP44, PLCC44, CA–BGA64
T89C51CC02	2.7–3.3 4.5–5.5	40	20	16K	2K	512	UART, CAN	10–bit ADC	3x16–bit PCA, WDT	PLCC28, SOIC28, TSSOP28, SOIC24
AT89C51CC03	2.7–3.3 4.5–5.5	40	34	64K	2K	2304	UART, CAN	10–bit ADC	3x16–bit PCA, WDT	VQFP44, PLCC44, CA–BGA64
AT89LP213	2.4–5.5	20	14	2K	–	128	TWI, SPI	–	2x16–bit WDT	TSSOP14, PDIP14
AT89LP214	2.4–5.5	20	12	2K	–	128	UART, TWI, SPI	–	2x16–bit WDT	TSSOP14, PDIP14
AT89LP216	2.4–5.5	20	14	2K	–	128	UART, TWI, SPI	–	2x16–bit WDT	SOIC16, PDIP16, TSSOP16
AT89LP2052	2.4–5.5	20	15	2K	2K	256K	UART, SPI	10–bit ADC	2x16–bit PWM, WDT	PDIP20, TSSOP20, SOIC20
AT89LP4052	2.4–5.5	20	15	4K	2K	256K	UART, SPI	10–bit ADC	2x16–bit PWM, WDT	PDIP20, TSSOP20, SOIC20

Перш ніж перейти до розгляду продукції фірми Temic, коротко опишемо МК типу MCS–52 та їх основні відмінності від MCS–51. Сімейство MCS–52 являє собою вдосконалене сімейство MCS–51 – з розширеними функціональними можливостями та підвищеною продуктивністю. МК типу MCS–52 програмно сумісні з сімейством MCS–51. В MCS–52 збільшено об`єм пам'яті програм на кристалі (в деяких аж до 64 Кбайт), введено додаткові регістри спеціальних функцій і нові режими роботи, підвищено захищеність програм від нелегального копіювання, лінії порту 1 використовуються в альтернативних режимах, розширено і виконано більш гнучкою систему переривань, розширено об’єм внутрішньої пам’яті даних. Сімейство MCS–52 навряд можна

розглядати як самостійне – для цих МК справедливо абсолютно все, що було раніше сказано для MCS–51. Але в той же час ці МК створюють «підсімейство» в сімействі MCS–51, оскільки між собою вони різняться лише наявністю чи відсутністю внутрішньої пам'яті програм і об'ємом цієї пам'яті, а їх нові (у зрівнянні з MCS–51) функціональні можливості досить великі.

Нижче наведено характеристики деяких мікроконтролерів фірми Temic (таблиця 1.2).

Таблиця 1.2 – Мікроконтролери фірми Atmel Wireless (Temic)

Назва	Корпуса	Пам'ять програм	Тактові частоти, МГц	Особливості
T89C51AC2	PLCC44, VQFP44, CA–BGA64	Флеш 32K	20, 40	Розширена пам'ять XRAM – 1K. EEPROM даних – 2K. Два DPTR. Режим X2 (6 тактів на машинний цикл). АЦП 8x10 біт,
T89C51RB2	DIP40, PLCC44, VQFP44,	Флеш 16K	30, 40	Розширена пам'ять XRAM – 1K. Інтерфейс SPI. Режим X2 для кожного периферійного блоку. Інтерфейс клавіатури.
T89C51RC2	DIP40, PLCC44, VQFP44,	Флеш 32K	30, 40	Розширена пам'ять XRAM – 1K. Інтерфейс SPI. Режим X2 для кожного периферійного блоку. Інтерфейс клавіатури.
T89C51RD2	DIP40, PLCC44, VQFP44, PLCC68, VQFP64	Флеш 64K	25, 40	Найбільша пам'ять програм в сімействі. Розширена пам'ять XRAM – 1K. Два DPTR.
T89C51IC2	PLCC44, VQFP44	Флеш 32K	30, 40	Розширена пам'ять XRAM – 1K. Інтерфейс SPI, i^2c. Інтерфейс клавіатури.

Всі перелічені в таблиці мікроконтролери мають можливість програмуватись внутрішньосхемно через UART. ОЗП даних в порівнянні зі стандартною архітектурою MCS–51 розширено на 1 Кбайт і містить 1280 байт. Об`єм флеш–пам'яті програм більший, ніж у кристалів Atmel. Мікроконтролери мають режим заборони видачі сигналу ALE для зменшення радіозавад. До складу периферійних пристроїв входить вартовий таймер.

1.4 Мікроконтролери Siemens (Infineon)

Однією із найавторитетніших компаній на ринку мікроконтролерів є Siemens. В результаті реорганізації підрозділ по випуску напівпровідників отримав свою власну назву – Infineon. Фірма Infineon випускає, разом з іншими цікавими мікросхемами, мікроконтролери сімейства C500. Архітектура C500 базується на MCS–52, але має більш розвинуту периферію. Характеристики деяких мікроконтролерів сімейства C500 приведено в таблиці 1.3. Специфіка мікроконтролерів Infineon визначається тим, що фірма Siemens спеціалізується на випуску апаратури промислової автоматики. 8–розрядне сімейство призначається, в першу чергу, для використання в інтелектуальних датчиках і нескладних приладах автоматики, які не потребують частого оновлення програмного забезпечення. Для таких пристроїв характерно використання в якості пам'яті програм маскового або однократно програмованого ПЗП (Mask ROM або OTP ROM). Більшість кристалів мають версії з обома типами пам'яті або без внутрішньої пам'яті програм. Більшість мікроконтролерів мають в своєму складі багатоканальні широтно–імульсні модулятори (ШІМ) і аналого–цифрові перетворювачі.

Таблиця 1.3 – Мікроконтролери фірми Infineon (Siemens)

Назва	Корпуса	Пам'ять	ОЗП, байт	ШІМ, кан.	АЦП, кан. х біт	Такт. част., МГц	Особливості
C501G	DIP40, PLCC44, MQFP44	8K OTP	256	–	–	24	Контролер загального користування
C504	MQFP44	Немає або 16K ROM/OTP	512	6	8x10	24	Модуль керування двигуном пост. струму
C513AO	PLCC44	Немає або 16K ROM/OTP	512	–	–	16	Низьке енергоспоживання і електромагнітне випромінювання
C508	SDIP64, MQFP64	32K ROM/OTP	1280	6	8x10	20	Модуль керування двигуном пост. току, множник тактової частоти

Продовження таблиці 1.3

Назва	Корпуса	Пам’ять	ОЗП, байт	ШІМ, кан.	АЦП, кан. х біт	Такт. част., МГц	Особливості
C505L	MQFP80	32K OTP	512	4	8x10	20	Драйвер ЖКІ, годинник реального часу
C505A	MQFP44	32K OTP	1280	4	8x10	20	Низьке енергоспоживання і електромагнітне випромінювання
C505CA	MQFP44	32K ROM/OTP або 16K ROM	1280	4	8x10	20	Низьке енергоспоживання і електромагнітне випромінювання
C515	PLCC68	Немає	256	4	8x8	24	Сумісний з SAB 80C515
C515C	MQFP80	Немає або 64K ROM/OTP	2304	4	8x10	10	Низьке енергоспоживання і електромагнітне випромінювання
C517A	PLCC84	Немає	2304	21	12x8	18	Сумісний з SAB 80C517A
C509	MQFP100	Немає	3328	29	12x10	16	Завантажувач для зовнішньої флеш–пам’яті програм
C541	PLCC44	8K OTP	256	–	–	12	USB – функція

1.5 Мікроконтролери Philips

Велику номенклатуру мікроконтролерів з ядром MCS–51 випускає фірма Philips. Продукцію фірми орієнтовано, в першу чергу, на ринок побутової електроніки. Більшість кристалів мають інтерфейс I^2C. I^2C – це послідовна шина, яку запропоновано фірмою Philips в якості стандарту обміну в недорогих низькошвидкісних системах. Характеристики деяких мікроконтролерів Philips наведено в таблиці 1.4. Буква х в назві мікросхеми означає, що даний кристал може поставлятися з різними типами пам'яті програм:

0 – без внутрішньої пам'яті;

3 – з масковим ПЗП (запрограмованим на замовлення);

7 – з ППЗП (EPROM);

9 – з флеш–пам'яттю.

Фірма Philips, так само, як і фірма Atmel, випускає мікроконтролери в корпусах з малим числом виводів, деякі з яких також представлено в таблиці 1.4.

Таблиця 1.4 – Мікроконтролери фірми Philips

Назва	Корпус	Пам'ять програм	Тактова частота, МГц	Особливості
P8xC552 x=0,3,7	PLCC68, QFP80	8K	16 МГц, 24 МГц	АЦП 8x10 біт, I^2C
P8xC562 x=0,3	PLCC68	8K	16 МГц, 24 МГц	АЦП 8x8 біт, I^2C
P8xC557 x=0,3,9	QFP80	32K	16 МГц	ОЗП 1К, АЦП 8x10 біт, ШІМ, I^2C, множник тактової частоти
P80C31/ P80C32	DIP40, PLCC44, QFP44	–	16 МГц, 33 МГц	ОЗП128/256 б. Режим заборони видачі ALE
P8xC51FA/FB/FC x=0,3,7	DIP40, PLCC44, QFP44	8/16/32K	16 МГц, 33 МГц	ОЗП 256 б. Додатковий таймер.
P8xC51RA+/RB+/RC+ x=0,3,7	DIP40, PLCC44, QFP44	8/16/32K	16 МГц, 33 МГц	ОЗП 512 б. Додатковий таймер, вартовий таймер.
P8xC51RD+ x=0,3,7	DIP40, PLCC44, QFP44	64K	16 МГц, 33 МГц	ОЗП 1024 б. Додатковий таймер, вартовий таймер.
P8xC748 x=7,9	DIP24, SOIC24, PLCC28	2K	16 МГц	Зменшений корпус, ОЗП 64 б.
P8xC752 x=7,9	DIP24, SOIC24, PLCC28	2K	16 МГц	Зменшений корпус, ОЗП 64 б, АЦП 5x8 біт, i^2c.

1.6 Мікроконтролери Analog Devices

Мікроконтролери Analog Devices не є стандартними виробами, однак кількість проданих мікросхем та їх велика популярність не дозволяють залишити без уваги продукцію цієї фірми в даному огляді. Технічні дані деяких мікроконтролерів наведені в таблиці 1.5.

Таблиця 1.5 – Мікроконтролери фірми Analog Devices

Тип	ADC, каналів/біт	DAC, каналів/біт	Flash програм	EEPROM даних	RAM	Корпус	Додатково
ADuC812	8/12	2/12	8K	640b	256b	PQFP52, CSP56	5 µs ADC Conversion
ADuC814	6/12	2/12	8K	640b	256b	TSSOP28	Small, Low–Cost, Low–Power
ADuC816	2/16	1/12	8K	640b	256b	PQFP52, CSP56	Programmable Gain Input

Продовження таблиці 1.5

Тип	ADC, каналів/біт	DAC, каналів/біт	Flash програм	EEPROM даних	RAM	Корпус	Додатково
ADuC824	1/24+1/16	1/12	8K	640b	256b	PQFP52, CSP56	Pin–Compatible Upgrade to ADuC816
ADuC831	8/12	2/12 + 2 PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC812
ADuC832	8/12/	2/12–Bit + 2 PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC812 plus PLL
ADuC834	1/24+1/16	Single 12–Bit + Dual PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC824
ADuC836	2/16	1/12 + 2 PWM	62K	4K	256+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC816
ADuC841	8/12	2/12 + 2 PWM	62K	4K	256+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC812 Fast 8052 Core
ADuC842	8/12	2/12 + 2 PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC812 plus PLL Fast 8052 Core
ADuC844	1/24+1/16	Single 12–Bit + Dual PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC824 Fast 8052 Core
ADuC845	10/24+1/24	Single 12–Bit + Dual 16–bit + Dual PWM 16bit	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC824 Fast 8052 Core
ADuC846	2/16	1/12 + 2 PWM	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC816 Fast 8052 Core
ADuC847	10/24	Dual 16–bit + Dual PWM 16bit	62K	4K	256b+2K	PQFP52, CSP56	"Big Memory" Upgrade to ADuC816 Fast 8052 Core

Analog Devices є одним з найбільших світових виробників АЦП і ЦАП; дослідницькі центри фірми вносять величезний вклад у розвиток цієї галузі. Багато рішень від Analog Devices стали стандартами де–факто, аналоги випускаються багатьма фірмами.

Analog Devices вийшли на ринок 8–розрядних мікроконтролерів, запропонувавши сімейство MicroConverter. У цих виробах об'єднано на одному кристалі найважливіші компоненти систем збору і обробки інформації – процесорне ядро MCS–52 і блок введення–виведення аналогової інформації, що включає багатоканальні ЦАП і АЦП. Ідея такого

об'єднання не нова, однак фірмі Analog Devices вдалося розмістити на кристалі з процесором дійсно високоякісний АЦП, з роздільною здатністю 12 біт і більше, з функціями калібрування і вимірювання температури.

1.7 Мікроконтролери Dallas Semiconductor

Особливістю мікроконтролерів фірми Dallas Semiconductor є змінена часова сітка. Машинний цикл в даних мікропроцесорах займає 4 такти замість 12 у класичного 80C51. При цьому збережено повну сумісність зі стандартною архітектурою на рівні інструкцій. Мікроконтролери Dallas знаходять застосування в задачах, які потребують підвищеної швидкості обробки. Характеристики деяких мікроконтролерів фірми наведено в таблиці 1.6.

Таблиця 1.6 – Мікроконтролери фірми Dallas

Назва	Корпуса	Пам’ять програм	Додатковий ОЗП	Кількість послідовних портів	Тактова частота	Особливості
DS80C310	DIP40, PLCC44, TQFP44	–	–	1	33 МГц	
DS80C320/323	DIP40, PLCC44, TQFP44	–	–	2	33 МГц, 18 МГц	
DS80C390	PLCC68, TQFP64	До 4K ОЗП	до 4K	2	40 МГц	Інтерфейс CAN 2.0B, математичний співпроцесор, перерозподіл ОЗП між областями даних і команд
DS80CH11	TQFP128	–	256b	1	33 МГц	ШІМ, АЦП 8x10 біт
DS8xC520	DIP40, PLCC44, TQFP44	16K ROM або EPROM	1K	2	33 МГц	
DS8xC530	PLCC52, TQFP52	16K ROM або EPROM	1K	2	33 МГц	Годинник реального часу

Багато мікроконтролерів мають розширений ОЗП, звернення до якого відбувається так само, як до зовнішньої пам'яті. У регістрову модель мікропроцесора включено другий покажчик зовнішньої пам'яті DPTR.

Характерною особливістю периферії мікроконтролерів Dallas є наявність на одному кристалі двох послідовних портів.

1.8 Склад та основні характеристики мікроконтролера ADuC847

ADuC847 є покращеною модифікацією мікроконтролера ADuC834 і функціонально близький до мікроконтролера ADuC845. ADuC847 має поліпшену, 12.58MIPs, версію ядра МК 8052. У порівнянні з ADuC845 ADuC847 містить більшу кількість каналів аналогового введення, але в ньому відсутні ЦАП і додатковий АЦП. Виріб має всі функції ADuC834, стандартне 12–циклове ядро замінено одноцикловим з продуктивністю 12.58MIPs.

Загальний опис

ADuC847 є закінченим контролером для інтелектуальних датчиків, що включає в себе сигма–дельта АЦП високого розширення, гнучкий вхідний мультиплексор на 10/8 каналів, швидкий 8–розрядний мікроконтролер і вбудовану Flash/EEPROM – пам'ять програм і даних.

До складу АЦП включено вхідний мультиплексор, датчик температури і підсилювач з програмованим коефіцієнтом передачі (PGA), який дозволяє працювати з сигналами низького рівня, які знімаються безпосередньо з датчиків. АЦП з вбудованим фільтром і програмованим вихідним потоком даних призначено для вимірювання низькочастотних сигналів у широкому динамічному діапазоні напруг таких, як сигнали з зважувальних пристроїв, сигнали з датчиків деформації, з датчиків тиску або температури.

Пристрій працює від кварцового резонатора 32кГц, а висока частота 12.58МГц виробляється системою фазового автопідстроювання частоти (ФАПЧ). Висока частота проходить через програмно–керований дільник, з якого знімається частота для роботи мікропроцесорного ядра.

Мікропроцесорне ядро є оптимізованим одноцикловим ядром 8052, що дає продуктивність до 12.58MIPS, при виконанні команд, сумісних з МК 8051.

Пристрій містить 62Кбайт внутрішньої Flash–пам'яті програм, 4Кбайт внутрішньої EEPROM пам'яті даних і 2304 байт внутрішньої пам'яті даних з довільним доступом (SRAM).

«Зашите» на етапі виробництва програмне забезпечення дозволяє робити завантаження програм в пристрій через послідовний порт (UART), а також виконувати налагодження прикладних програм системи через єдиний зовнішній вхід EA. Розробка виробів на основі ADuC847 підтримується недорогою системою QuickStart TM (апаратура і програми).

Робота таймера

У класичному МК 8052 всі таймери інкрементуються на +1 під час кожного машинного циклу. Оскільки в ADuC847 один машинний цикл дорівнює одному періоду тактової частоти, таймери будуть інкрементуються з частотою, яка дорівнює тактовій частоті ядра.

Сигнал ALE

Частота вихідного сигналу ALE мікроконтролера ADuC834 становить 1/6 тактової частоти ядра. У ядрі ADuC847 сигнал ALE виглядає наступним чином.

У разі одноциклової команди вихід ALE знаходиться у високому логічному стані протягом першої половини машинного циклу і в низькому – протягом другої половини. Тут частота на виході ALE дорівнює тактовій частоті ядра. Для команд, що містять два або більше циклів, вихід ALE знаходиться у високому логічному стані протягом першої половини машинного циклу і в низькому – протягом заключної половини.

Доступ до зовнішньої пам'яті

ADuC847 не підтримує доступ до зовнішньої пам'яті програм. При зверненні до зовнішньої пам'яті даних з довільним доступом буде потрібно

програмувати регістр EWAIT, щоб при виконанні команди MOVX були виконані додаткові машинні цикли. Це необхідно виконати через відмінності у часі доступу до внутрішньої і зовнішньої SRAM.

Більш детальна інформація про МК–р ADuC847 наведена у [12].

1.9 Склад та основні характеристики мікроконтролерів фірми Silicon Laboratories (Cygnal)

Фірма Silicon Laboratories [14] динамічно розвиває перелік мікроконтролерів сімейства MCS–51. Фірмі вдалося створити більш ніж 55 типів мікроконтролерів, що відрізняються продуктивністю, об'ємом Flash пам'яті програм, вбудованої оперативної пам'яті, характеристиками аналого–цифрових вузлів, типом корпусу та іншими параметрами. Деякі мікроконтролери, що випускаються фірмою, наведено в таблиці 1.7. Їх умовно розділено на 11 підсімейств.

Таблиця 1.7 – Мікроконтролери фірми Silicon Laboratories

Підсімейство	Тип	Основні параметри					
		Продуктив–ність MIPS	Flash ROM, K	RAM, byte	ADC, bit	Корпус (кількість виводів)	Примітка
C8051F0xx	C8051F000	20	32	256	12	64	
	C8051F001	20	32	256	12	48	
	C8051F002	20	32	256	12	32	
	C8051F005	25	32	2.25K	12	64	
	C8051F006	25	32	2.25K	12	48	
	C8051F007	25	32	2.25K	12	32	
	C8051F010	20	32	256	10	64	
	C8051F011	20	32	256	10	48	
	C8051F012	20	32	256	10	32	
	C8051F015	25	32	2.25K	10	64	
	C8051F016	25	32	2.25K	10	48	
	C8051F017	25	32	2.25K	10	32	
C8051F018/9	C8051F018	25	16	1.25K	10	48	
	C8051F019	25	16	1.25K	10	32	

Продовження таблиці 1.7

Сімейство	Тип	Основні параметри					
		Продуктив–ність MIPS	Flash ROM, K	RAM, byte	ADC, bit	Корпус (кількість виводів)	Примітка
C8051F02x	C8051F020	25	64	4.25K	12+8	100	
	C8051F021	25	64	4.25K	12+8	64	
	C8051F022	25	64	4.25K	10+8	100	
	C8051F023	25	64	4.25K	10+8	64	
C8051F04x	C8051F040	25	64	4.352K	12+8	100	CAN 2.0B
	C8051F041	25	64	4.352K	12+8	64	CAN 2.0B
	C8051F042	25	64	4.352K	10+8	100	CAN 2.0B
	C8051F043	25	64	4.352K	10+8	64	CAN 2.0B
	C8051F044	25	64	4.352K	10+8	100	CAN 2.0B
	C8051F045	25	64	4.352K	10+8	64	CAN 2.0B
	C8051F046	25	32	4.352K	10+8	100	CAN 2.0B
	C8051F047	25	32	4.352K	10+8	64	CAN 2.0B
C8051F06x	C8051F060	25	64	4.352K	16+10	100	CAN+DMA
	C8051F061	25	64	4.352K	16+10	64	CAN+DMA
	C8051F062	25	64	4.352K	16+10	100	CAN+DMA
	C8051F063	25	64	4.352K	16+10	64	CAN+DMA
C8051F12x	C8051F120	100	128	8.25K	12+8	100	
	C8051F121	100	128	8.25K	12+8	64	
	C8051F122	100	128	8.25K	10+8	100	
	C8051F123	100	128	8.25K	10+8	64	
	C8051F124	50	128	8.25K	12+8	100	
	C8051F125	50	128	8.25K	12+8	64	
	C8051F126	50	128	8.25K	10+8	100	
	C8051F127	50	128	8.25K	10+8	64	
C8051F13x	C8051F130	100	128	8.25K	10	100	
	C8051F131	100	128	8.25K	10	64	
	C8051F132	100	64	8.25K	10	100	
	C8051F133	100	64	8.25K	10	64	
C8051F2xx	C8051F206	25	8	1.25K	12	48	
	C8051F220	25	8	256	8	48	
	C8051F221	25	8	256	8	32	
	C8051F226	25	8	1.25K	8	48	
	C8051F230	25	8	256	–	48	
	C8051F231	25	8	256	–	32	
	C8051F236	25	8	1.25K	–	48	
C8051F30x	C8051F300	25	8	256	8	11	2% Osc
	C8051F300P	25	8	256	8	DIP14	2% Osc
	C8051F301	25	8	256	–	11	2% Osc
	C8051F302	25	8	256	8	11	
	C8051F303	25	8	256	–	11	
	C8051F304	25	4	256	–	11	
	C8051F305	25	3	256	–	11	
C8051F31x	C8051F310	25	16	1.28K	10	32	2% Osc
	C8051F311	25	16	1.28K	10	28	2% Osc
C8051F32x	C8051F320	25	16	2.3K	10	32	Osc + USB
	C8051F321	25	16	2.3K	10	28	Osc + USB

Продовження таблиці 1.7

Сімейство	Тип	Основні параметри					
		Продуктив–ність MIPS	Flash ROM, K	RAM, byte	ADC, bit	Корпус (кількість виводів)	Примітка
C8051F33x	C8051F330	25	8	768	10	20	Osc
	C8051F330P	25	8	768	10	DIP14	Osc
	C8051F331	25	8	768	–	20	Osc

При розробці програмного забезпечення для мікроконтролерів Cygnal можна використовувати стандартні x51–орієнтовані асемблери і компілятори мов високого рівня (C–51, PL/M–51, FORTRAN–51, PASCAL–51 та ін.), а також бібліотеки підпрограм, які створені різними колективами за десятиліття існування x51–сумісних мікроконтролерів. Ядро мікроконтролерів Cygnal (CIP–51) містить повний набір периферійних вузлів, стандартний для MCS–51. Залежно від типу підсімейства, ядро може містити 3, 4 або 5 таймерів–лічильників, один або два послідовних порти UART, як мінімум 256 байт вбудованої оперативної пам'яті, 128–байт регістрів спеціальних функцій SFR (Special Function Register). Ядро також забезпечує керування лініями портів введення/виведення, яких мікроконтролери різних підсімейств можуть мати від 1 до 8 (тобто від 8 до 64 ліній введення/виведення). Також перевагою цього ядра є вбудована апаратура налагодження. Ядро забезпечує безпосереднє керування аналоговими і цифровими підсистемами мікроконтролера через відповідні регістри SFR. Таким чином, ядро CIP–51 з одного боку забезпечує повну сумісність зі стандартним x51–сумісним ядром, з іншого, має значно ширші апаратні можливості за рахунок поповнення вбудованою цифровою і аналоговою периферією. Структурну схему ядра CIP–51 показано на рисунку 1.1.

Безсумніву, найважливішою перевагою ядра CIP–51 є вдосконалена конвеєрна архітектура, яка дозволяє значно збільшити продуктивність у

порівнянні зі стандартною x51 (8051) архітектурою. У мікроконтролерах зі стандартною архітектурою 8051 всі інструкції, за винятком MUL і DIV, виконувалися за 12 або 24 машинних тактів. При цьому максимальна тактова частота для більшості x51 сумісних мікроконтролерів становила 12–24 МГц, і лише деякі мікроконтролери могли працювати на вищих частотах.

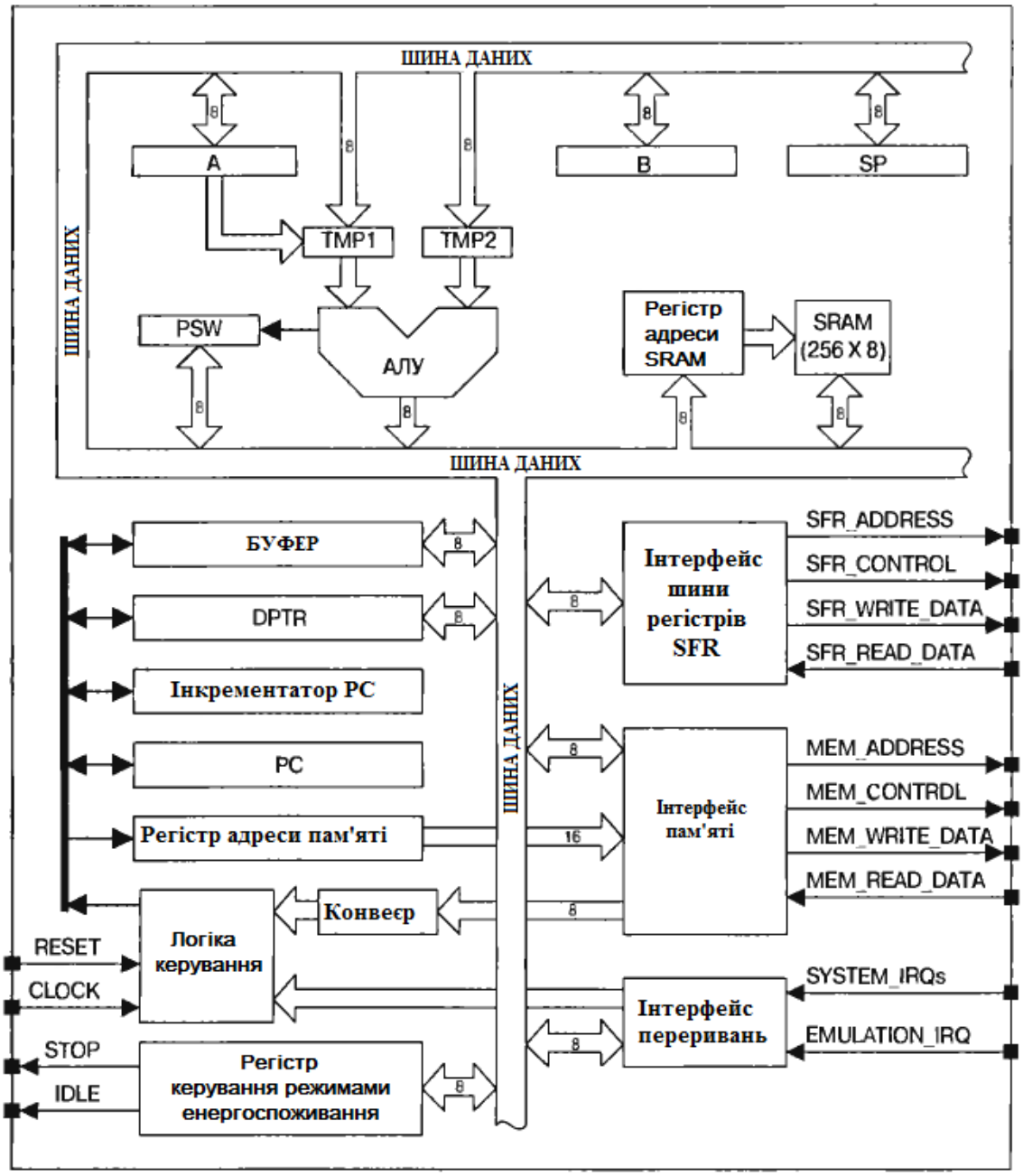


Рисунок 1.1 – Структура ядра CIP–51

Модернізоване ядро CIP–51 виконує 70% інструкцій за один або два машинних такти, і взагалі не має інструкцій, що виконуються більш ніж за вісім машинних тактів. При цьому, практично всі мікроконтролери фірми Silicon Laboratories (27 типів, 72%) можуть працювати при частоті тактового генератора 25МГц, шість мікроконтролерів працюють на частотах до 20МГц (16%), і чотири нових мікроконтролера сьомого сімейства (C8051F12x) функціонують на частотах до 100МГц. При цьому, відповідно розвивається пікова (гранична) продуктивність 25, 20 і 100 мільйонів інструкцій у секунду – MIPS (Million Instructions Per Second). Як приклад на рисунку 1.2 показано співвідношення граничної продуктивності деяких поширених мікроконтролерів в порівнянні з граничною продуктивністю мікроконтролерів фірми Silicon Laboratories при тактовій частоті 25МГц.

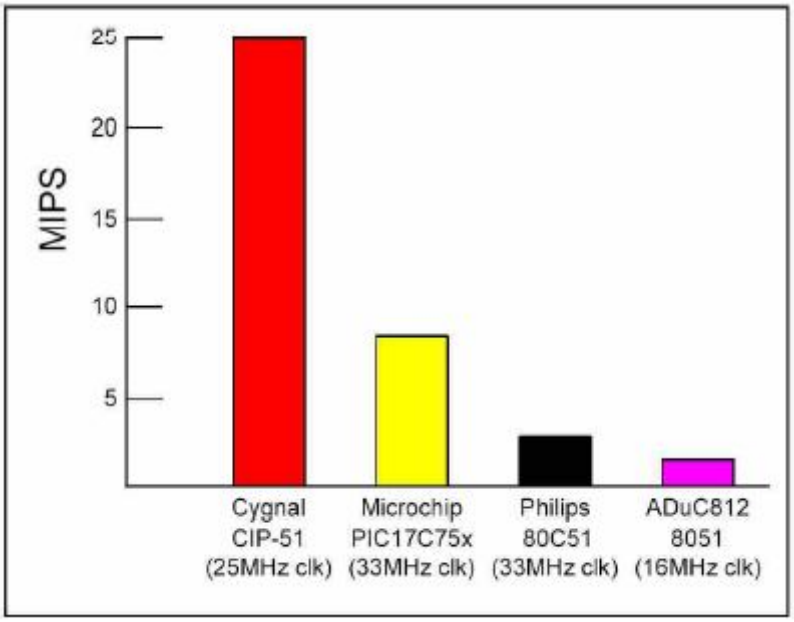


Рисунок 1.2 – Співвідношення граничної продуктивності деяких мікроконтролерів

Ще однією важливою перевагою мікропроцесорного ядра CIP–51 є наявність вбудованої апаратної підсистеми налагодження програмного

забезпечення. Зв'язок з підсистемою мікроконтролера здійснюється через послідовний інтерфейс JTAG, що відповідає специфікації IEEE 1149.1. При цьому, забезпечуються як режим внутрішньо системного програмування – ISP (In System Programarable), так і власне режим відлагодження. При програмуванні можливий запис, як усього масиву програми, так і модифікація окремих байтів. Природно, що вміст пам'яті програм може також читатися і звірятися з оригіналом. Вміст будь–якого байта програм може читатися або змінюватися з використанням інструкцій MOVC або MOVX, що також дозволяє здійснювати незалежне зберігання даних і оперативно їх модифікувати під керування програми.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Які основні характеристики мікроконтролерів сімейства МК–51?
2) Назвіть основні відмінності MCS–52 від MCS–51.
3) Чому мікроконтролери фірми Atmel набули популярності в країнах СНД?
4) Опишіть особливості мікроконтролерів таких фірм як:
    - Atmel;
    - Temic;
    - Siemens (Infineon);
    - Philips;
    - Analog Devices;
    - Dallas Semiconductor;
    - Silicon Laboratories (Cygnal).
5) Охарактеризуйте склад та основні характеристики мікроконтролера ADuC847.
6) Опишіть ядро CIP–51. Яка в нього структура?

ЛІТЕРАТУРА [4, 5, 14]

2 СТРУКТУРА ТИПОВОГО МІКРОКОНТРОЛЕРА МК51

2.1 Загальні відомості

Типовий МК AT89C51 [4–6] складається з таких основних функціональних вузлів (рисунок 2.1): блока керування і синхронізації; блока арифметико–логічного пристрою АЛП (англ. Arifmetic–Logic Unit); резидентної пам'яті даних РПД (англ. Resident Data Memory) об'ємом 128 байт; резидентної пам'яті програм РПП (англ. Resident Program Memory) об'ємом 4 Кбайт; блока переривань (англ. Interrupt System), таймерів (англ. Timer), послідовного порту (англ. Serial Port); чотирьох паралельних портів введення/виведення (англ. Parallel Port), які програмуються; схеми десяткової корекції вмісту акумулятора (СДКА); внутрішнього генератора тактових імпульсів ГТІ (англ. Oscillator); резидентної шини даних РШД (англ. Resident Data Bus) і групи регістрів:

А – акумулятор;

В – регістр розширення акумулятора;

Т1, Т2 – регістри тимчасового збереження операндів;

РСП (PSW) – регістр стану програми (прапорців);

РК (IR) – регістр команд;

ЛК (PC) – лічильник команд (програмний лічильник);

РПД (DPTR) – регістр–покажчик даних, що складається з 2–х частин: молодшої – DPL і старшої – DPH;

РПС (SP) – регістр–покажчик стеку;

РА (RAR) – регістр адреси;

РРТЛ (TMOD) – регістр режимів таймерів/лічильників;

РКСТ (TCON) – регістр керування–статусу таймерів/лічильників;

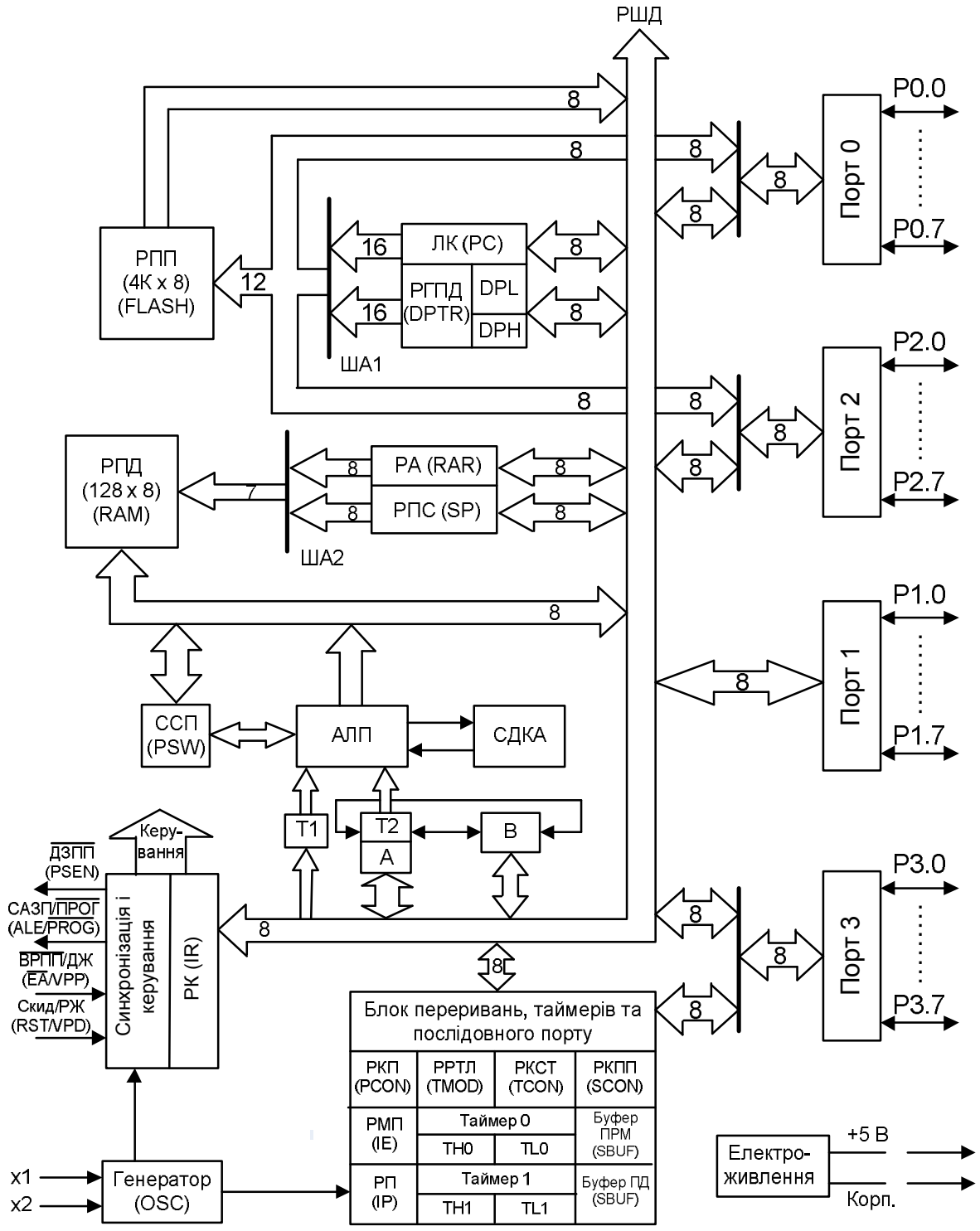


Рисунок 2.1 – Структурна схема МК AT89C51

РКПП (SCON) – регістр керування приймачем–передавачем послідовного порту;

SBUF (буфер ПРМ і *буфер ПД)* – буфери приймача і передавача послідовного порту;

РМП (IE) – регістр масок переривань;

РП (IP) – регістр пріоритетів переривань;

РКП (PCON) – регістр керування потужністю споживання енергії від джерела живлення.

2.2 Блок керування та синхронізації

Блок керування та синхронізації призначено для формування синхронізуючих і керуючих сигналів, що забезпечують координацію спільної роботи блоків МК у всіх допустимих режимах їх роботи.

До складу блока керування входять: пристрій формування часових інтервалів, логіка введення/виведення, регістр команд, дешифратор команд, ПЛМ і логіка керування контролером.

Пристрій формування часових інтервалів призначено для формування і видачі внутрішніх синхросигналів: станів, фаз і циклів. Кількість машинних циклів (англ. machine cycle) визначає тривалість виконання команд. Практично всі команди МК виконуються за один або два машинних цикли, крім команд множення MUL A, B і ділення DIV A, B, тривалість виконання яких складає чотири машинних цикли. Машинний цикл має фіксовану тривалість і містить шість станів (англ. state) S1–S6, кожен з яких складається з двох часових інтервалів, які називають фазами (англ. phase) P1 і P2.

Тривалість фази дорівнює періоду зовнішнього сигналу BQ, що є первинним сигналом синхронізації МК. Сигнал BQ формується або вмонтованим тактовим генератором МК (при підключенні до її виводів 18

(BQ2) і 19 (BQ1) кварцового резонатора або LC–ланцюжка), або зовнішнім джерелом тактових сигналів.

Схему підключення кварцового резонатора до виводів BQ2 і BQ1 показано на рисунку 2.2.

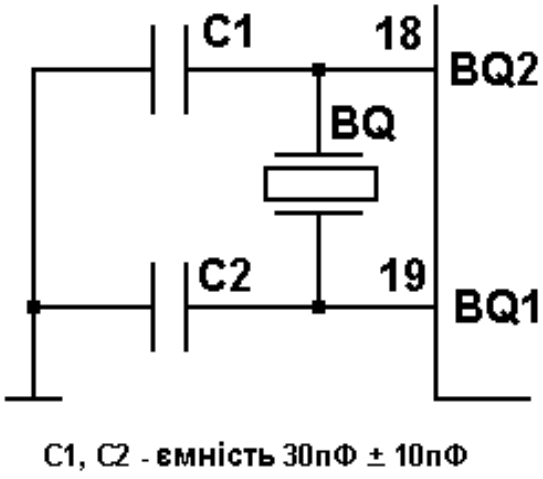


Рисунок 2.2 – Схема підключення кварцового резонатора

Рисунок 2.3 ілюструє формування машинних циклів в МК. Всі машинні цикли однакові, складаються з 12 періодів сигналу BQ, починаються фазою P1S1 і закінчуються фазою P2S6.

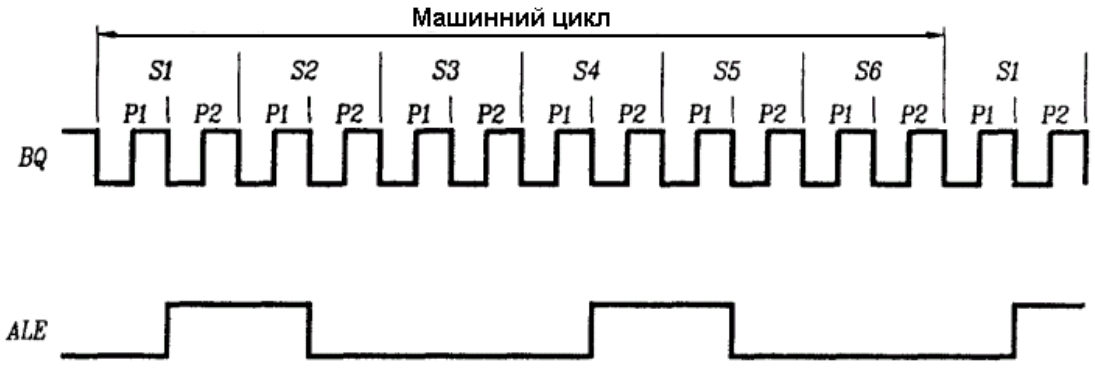


Рисунок 2.3 – Діаграма формування машинних циклів МК

Двічі за один машинний цикл формується сигнал дозволу фіксації адреси зовнішньої пам`яті ALE (англ. Address Latch Enable), що видається

на однойменний вивід. Якщо, наприклад, зовнішня частота f_{BQ} = 12 МГц, то тривалість машинного циклу $T_{МЦ}$ = 1 мкс.

В *регістр команд (РК),* який програмно не доступний, пересилається з пам'яті програм код операції чергової команди, що виконується. Дешифратор команд декодує код операції та ідентифікує тип команди, що підлягає виконанню.

Після цього з програмованої логічної матриці (ПЛМ) мікроконтролера викликається відповідна послідовність керуючих сигналів для виконання команди.

2.3 Блок арифметико–логічного пристрою

Блок арифметико–логічного пристрою (АЛП) забезпечує виконання арифметичних і логічних операцій, а також операцій логічного зсуву, скидання, встановлення і т.ін.

Блок АЛП складається з регістрів тимчасового збереження операндів Т1, Т2, ПЗП констант, суматора, додаткового регістра (регістра В), акумулятора та регістра стану програми.

Регістри тимчасового збереження операндів Т1, Т2 – 8–розрядні, які призначено для прийому і збереження операндів на час виконання операцій над ними. Вони програмно не доступні.

ПЗП констант забезпечує створення коду коригування при двійково–десятковому представленні даних, коду маски при бітових операціях і коду констант.

Паралельний 8–розрядний суматор являє собою схему комбінаційного типу з послідовним перенесенням, яку призначено для виконання арифметичних операцій додавання, віднімання і логічних операцій додавання, множення, порівняння тощо.

Регістр B – 8–розрядний регістр, який використовується під час операцій множення і ділення. Для інших інструкцій він може розглядатися як додатковий регістр користувача.

Акумулятор являє собою 8–розрядний регістр, який призначено для прийому і збереження результату, який отримано при виконанні арифметично–логічних операцій або операцій пересилання.

Регістр стану програми (PSW) призначено для збереження інформації про стан АЛП при виконанні програми. Позначення розрядів регістра PSW і призначення його розрядів наведено відповідно в таблицях 2.1 і 2.2.

Таблиця 2.1 – Позначення розрядів регістра PSW

Біти	7	6	5	4	3	2	1	0
Позначення	CY	AC	F0	RS1	RS0	OV	–	P

Таблиця 2.2 – Призначення окремих розрядів регістра PSW

Біти	Найменув.	Призначення бітів	Доступ до біта
7	CY	*Прапорець перенесення*. Змінюється під час виконання деяких арифметичних і логічних інструкцій	апаратно або програмно
6	AC	*Прапорець додаткового перенесення*. Апаратно встановлюється/скидається під час виконання інструкцій додавання або віднімання для зазначення перенесення або позики в біті 3 при утворенні молодшої тетради результату (D0–D3)	апаратно або програмно
5	F0	*Прапорець 0*. Прапорець стану, який визначається користувачем	програмно

Продовження таблиці 2.2

Біти	Найменув.		Призначення бітів	Доступ до біта
4	RS1		*Покажчик банку робочих регістрів РПД*	програмно
3	RS0		*Покажчик банку робочих регістрів РПД*	програмно
	RS1	RS0		
	0	0	Банк 0 з адресами (00H–07H)	
	0	1	Банк 1 з адресами (08H– 0FH)	
	1	0	Банк 2 з адресами (10H– 17H)	
	1	1	Банк 3 з адресами (18H–1FH)	
2	OV		*Прапорець переповнення*. Апаратно встановлюється/скидається під час виконання арифметичних інструкцій для зазначення стану переповнення розрядної сітки зі знаком	апаратно або програмно
1	–		*Резервний*. Містить тригер, доступний для запису ("0" і "1") і читання, який не рекомендується використовувати	програмно
0	P		*Біт парності*. Апаратно скидається/ встановлюється в кожному циклі інструкцій для зазначення парної/непарної кількості розрядів акумулятора, що знаходяться в стані "1"	апаратно або програмно

2.4 Резидентна пам'ять даних

Резидентну пам'ять даних РПД призначено для прийому, збереження і видачі інформації, яка використовується в процесі виконання програми. Пам'ять даних ділиться на резидентну (внутрішню) – РПД, і зовнішню –

ЗПД. До складу вузла РПД, який наведено на рисунку 2.1, входить ОЗП (SRAM) ємністю 128 байт і дешифратор адреси. Керують роботою РПД два регістри: РА (RAR) – регістр адреси; РПС (SP) – покажчик стеку.

Регістр адреси ОЗП (РА) призначено для прийому і збереження адреси комірки пам'яті, яка обирається за допомогою дешифратора і може містити як біт, так і байт інформації.

ОЗП являє собою 128 8–розрядних регістрів, які призначено для прийому, збереження і видачі різної оперативної інформації. 16 із цих регістрів допускають пряму бітову адресацію.

На рисунку 2.4 наведено розподіл адресного простору РПД і область бітів, що адресуються прямо.

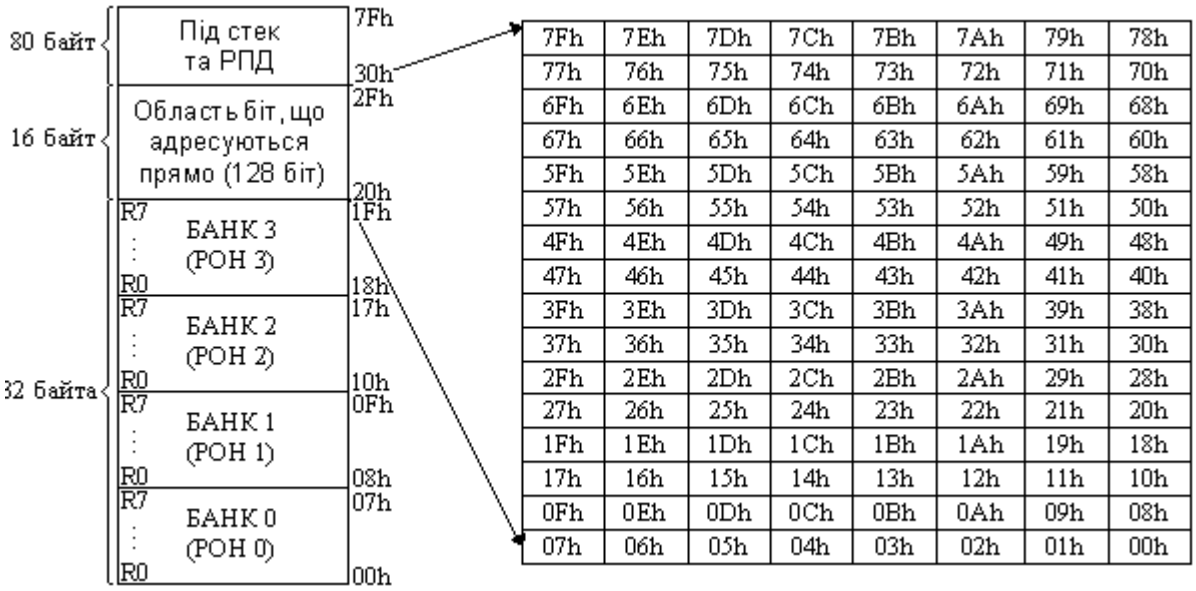


Рисунок 2.4 – Розподіл адресного простору РПД і область бітів, які адресуються прямо

Покажчик стеку являє собою 8–розрядний регістр, який призначено для прийому і збереження адреси комірки стеку. При виконанні команд LCALL та ACALL вміст покажчика стеку збільшується на 2. При виконанні команд RET та RETI вміст покажчика стеку зменшується на 2.

При виконанні команди PUSH direct вміст покажчика стеку збільшується на 1. При виконанні команди POP direct вміст покажчика стеку зменшується на 1. Після скидання в покажчику стеку встановлюється адреса 07H, що відповідає початку стеку з адресою 08H.

Більш докладно організацію пам'яті даних, яка використовується у цьому МК, розглянуто у [3,4].

2.5 Резидентна пам'ять програм

Резидентну пам'ять програм РПП призначено для збереження програм. Вона має окремий від пам'яті даних адресний простір об'ємом до 64 Кбайт, причому, для мікросхеми AT89C51 частину пам'яті програм з адресами 0000H–0FFFH розташовано на кристалі МК. Пам'ять програм РПП мікросхеми AT89C51 (рисунок 2.1), що розташована на кристалі, складається з 12–розрядного дешифратора та FLASH–ПЗП ємністю 4Кх8 біт. Запис програм у ПЗП відбувається під час розробки мікропроцесорної системи.

Якщо на вивід МК ВРПП (DEMA) подається напруга живлення U_{CC} (логічна 1), то звернення до зовнішньої пам'яті програм відбувається автоматично при видачі лічильником команд адреси, що перевищує 0FFFH. Якщо адреса знаходиться в межах 0000H–0FFFH, звернення відбувається до пам'яті програм, яка розташована на кристалі (резидентної пам'яті програм).

Якщо на вивід МК ВРПП подається сигнал "лог. 0", то внутрішня пам'ять програм відключається, і, починаючи з адреси 0000H, всі звернення виконуються до зовнішньої пам'яті програм.

Для формування поточної 16–розрядної адреси пам'яті програм служить лічильник команд (програмний лічильник) – ЛК (PC). 12 молодших розрядів цього регістра використовуються при адресації комірок РПП об'ємом 2^{12} = 4 Кбайт.

Більш докладно організацію пам'яті програм мікропроцесорних систем, в яких застосовуються даний МК, розглянуто у [3,4].

2.6 Блок переривань

МК AT89C51 має систему переривань із п'ятьма векторами (адресами підпрограм обробки переривань) і двома рівнями пріоритетів, які задаються програмно. Джерелами переривань виступають: два зовнішніх переривання, що надходять через порт 3; два переривання від переповнення таймерів/лічильників T/CNT0 і T/CNT1 та переривання при завершенні передачі або прийому даних при обміні через послідовний порт.

Для програмування і керування роботою системи переривань використовують два регістри: РМП (IE) – регістр масок переривань і РП (IP) – регістр пріоритетів переривань, а також чотири молодших біти регістра РКСТ (таблиці 2.3, 2.4).

Регістр пріоритетів переривань (IP) призначено для встановлення рівня пріоритету переривання для кожного з п'яти джерел переривань. Позначення розрядів регістра IP показано в таблиці 2.3, а їхнє призначення описано нижче.

Таблиця 2.3 – Позначення розрядів регістра IP

Біти	7	6	5	4	3	2	1	0
Позначення	X	X	X	PS	PT1	PX1	PT0	PX0

Призначення розрядів регістра IP:

- *PX0 (IP0)* – встановлення рівня пріоритету переривання від зовнішнього джерела $\overline{\text{INT0}}$;
- *PT0 (IP1)* – встановлення рівня пріоритету переривання від T/C0;

– *PX1 (IP2)* – встановлення рівня пріоритету переривання від зовнішнього джерела $\overline{\text{INT1}}$;
– *PT1 (IP3)* – встановлення рівня пріоритету переривання від Т/С1;
– *PS (IP4)* – встановлення рівня пріоритету переривання від послідовного порту;
– *X (IP7, IP6, IP5)* – резервний розряд.

Наявність у розрядах 0…4 регістра IP сигналу "логічна 1" встановлює для відповідного джерела високий рівень пріоритету, а наявність у цих розрядах сигналу "логічний 0" – низький рівень пріоритету. При читанні резервних розрядів їх значення не визначено. Користувач не повинен записувати "1" у резервні розряди, тому що вони зарезервовані для подальшого розширення сімейства МК–51.

Регістр дозволу переривань (IE) призначено для дозволу або заборони переривань від відповідних джерел. Позначення розрядів регістра IE показано в таблиці 2.4, а їхнє призначення описано нижче.

Таблиця 2.4 – Позначення розрядів регістра IE

Біти	7	6	5	4	3	2	1	0
Позначення	EA	X	X	ES	ET1	EX1	ET0	EX0

Призначення розрядів регістра IE:

– *EA (IE7)* – керування всіма джерелами переривань одночасно. Якщо EA = 0, то всі переривання заборонено. Якщо EA = 1, то використовуються індивідуальні дозволи переривань EX0, ET0, EX1, ET1, ES;
– *X (IE6, IE5)* – резервний розряд;

– *ES (IE4)* – керування перериванням від послідовного порту: ES = 1 – дозвіл, ES = 0 – заборона;
– *ET1 (IE3)* – керування перериванням від Т/С1: ET1 = 1 – дозвіл, ET1 = 0 – заборона;
– *EX1 (IE2)* – керування перериванням від зовнішнього джерела $\overline{\text{INT1}}$: EX1 = 1 – дозвіл, EX1 = 0 – заборона;
– *ET0 (IE1)* – керування перериванням від Т/С0: ET0 = 1 – дозвіл, ET0 = 0 – заборона;
– *EX0 (IE0)* – керування перериванням від зовнішнього джерела $\overline{\text{INT0}}$: EX0 = 1 – дозвіл, EX0 = 0 – заборона.

При читанні резервних розрядів їх значення не визначено. Користувач не повинен записувати сигнал "лог. 1" у резервні розряди, тому що їх зарезервовано для подальшого розширення сімейства МК–51.

Блок переривань містить також *схему логічної обробки переривань*, яка здійснює пріоритетний вибір запиту переривання, скидання його прапорця та ініціює формування апаратно реалізованої команди переходу на підпрограму обслуговування переривання LCALL.

Схема формування вектора переривання формує двобайтні адреси підпрограм обслуговування переривань в залежності від джерел переривання, які наведено в таблиці 2.5.

Більш докладно підсистему переривань описано у [4…6].

2.7 Блок таймерів–лічильників

Таймери/лічильники (Т/Л) призначено для лічби зовнішніх подій, для формування програмно керованих часових затримок і виконання функцій мікроконтролера, які задають часові інтервали.

Таблиця 2.5 – Джерела переривань і адреси підпрограм, що їх обслуговують

Джерело переривання	Вектор переривання
Зовнішнє переривання $\overline{INT0}$	0003H
Таймер/лічильник Т/С0	000BH
Зовнішнє переривання $\overline{INT1}$	0013H
Таймер/лічильник Т/С1	001BH
Послідовний порт	0023H

До складу блоку Т/Л входять:

- два 16–розрядних регістри Т/Л0 і Т/Л1;
- 8–розрядний регістр режимів Т/Л (TMOD);
- 8–розрядний регістр керування (TCON);
- схема інкременту;
- схема фіксації $\overline{INT0}$, $\overline{INT1}$, T0, T1;
- схема керування прапорцями;
- логіка керування Т/Л.

Два 16–розрядних регістри Т/Л0 і *Т/Л1* виконують функцію зберігання вмісту лічби. Кожен з них складається з пари восьмирозрядних регістрів, відповідно TH0, TL0 і TH1, TL1. Причому регістри TH0, TH1 – старші, а регістри TL0, TL1 – молодші 8 розрядів. Кожен із восьмирозрядних регістрів має свою адресу і може бути використаний як РЗП, якщо Т/Л не використовуються (біт TR0 для Т/Л0 і біт TR1 для Т/Л1 у регістрі керування TCON дорівнюють "0").

Стартове значення регістрів Т/Л задається програмно. В процесі лічби вміст регістрів Т/Л інкрементується. Ознакою закінчення лічби, як правило, виступає переповнення регістра Т/Л, тобто перехід його вмісту зі

стану "всі одиниці" у стан "усі нулі". Всі регістри TH0, TH1, TL0, TL1 доступні для читання, і, при необхідності, контроль досягнення необхідного значення може виконуватися програмно.

Регістр режимів Т/Л (TMOD) призначено для прийому і збереження коду керування, який визначає:

– один із 4–х можливих режимів роботи кожного Т/Л;

– роботу в якості таймерів або лічильників зовнішніх подій;

– керування Т/Л від зовнішнього виводу.

Позначення розрядів регістра TMOD наведено в таблиці 2.6, а призначення розрядів – у таблиці 2.7.

При роботі в якості таймера вміст регістра Т/Л інкрементується в кожному машинному циклі, тобто Т/Л є лічильником машинних циклів МК. Оскільки машинний цикл складається з 12 періодів частоти синхронізації МК f_{BQ}, то частота лічби в даному випадку дорівнює $f_{BQ}/12$.

При роботі Т/Л в якості лічильника зовнішніх подій вміст регістра Т/Л інкрементується у відповідь на перехід із стану "лог. 1" у стан "лог. 0" сигналу на лічильному вході МК (вивід Т0 для Т/Л0 і вивід Т1 для Т/Л1). Лічильні входи апаратно перевіряються у фазі S5 P2 кожного машинного циклу.

Коли перевірки показують високий рівень на лічильному вході в одному машинному циклі і низький рівень в іншому машинному циклі, регістр Т/Л інкрементується. Нове (інкрементоване) значення заноситься в регістр Т/Л у фазі S3 P1 машинного циклу, що безпосередньо іде за тим, у якому було виявлено перехід із стану "лог. 1" у стан "лог. 0" на лічильному вході МК. Оскільки для розпізнавання такого переходу потрібно два машинних цикли (24 періоди частоти синхронізації МК f_{BQ}), то максимальна частота лічби Т/Л в режимі лічильника дорівнює $f_{BQ}/24$.

Таблиця 2.6 – Позначення розрядів регістра TMOD

Біти	7	6	5	4	3	2	1	0
Позн.	GATE1	$C/\overline{T}1$	M1.1	M0.1	GATE0	$C/\overline{T}0$	M1.0	M0.0

Таблиця 2.7 – Призначення розрядів регістра TMOD

<table>
<tr><th>Біти</th><th>Найменув.</th><th colspan="3">Призначення бітів</th><th>Примітка</th></tr>
<tr><td rowspan="6">0–1
4–5</td><td rowspan="6">M0–M1</td><td colspan="3">Визначають один із 4–х режимів роботи, окремо для Т/Л1 і Т/Л0.</td><td rowspan="6">Усі біти встановлюються програмно;
біти 0–3 визначають режим роботи Т/Л0,
біти 4–7 визначають режим роботи Т/Л1.</td></tr>
<tr><td>M1</td><td>M0</td><td>Режим</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>3</td></tr>
<tr><td>2, 6</td><td>$C/\overline{T}0$
$C/\overline{T}1$</td><td colspan="3">Визначають роботу в якості:
C/T0, C/T1 = 0 – таймера
C/T0, C/T1 = 1 – лічильника зовнішніх подій</td><td rowspan="2"></td></tr>
<tr><td>3, 7</td><td>GATE</td><td colspan="3">Дозволяє керувати таймером від зовнішнього виводу ($\overline{INT0}$ – для Т/Л0, $\overline{INT1}$ – для Т/Л1).
GATE = 0 – керування заборонено
GATE = 1 – керування дозволено</td></tr>
</table>

Щоб рівень сигналу на лічильному вході було гарантовано зафіксовано, він повинен залишатися незмінним протягом як мінімум одного машинного циклу.

Регістр керування (TCON) призначено для прийому і збереження коду керуючого слова. Позначення розрядів регістра TCON наведено в таблиці 2.8, а призначення розрядів – у таблиці 2.9.

Таблиця 2.8 – Позначення розрядів регістра TCON

Біти	7	6	5	4	3	2	1	0
Позначення	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Таблиця 2.9 – Призначення розрядів регістра TCON

Біти	Найменув.	Призначення бітів	Примітка
6 4	TR1 TR0	Біти включення Т/Л, окремо для Т/Л0 і Т/Л1. TR = 0 – виключений, TR = 1 – включений.	Біти встановлюються і скидаються програмно. Доступні за читанням.
7 5	TF1 TF0	Прапорці переповнення Т/Л.	Біти скидаються і встановлюються апаратно і програмно. Доступні за читанням.
2 0	IT1 IT0	Біти, що визначають вид переривання за входами INT1, INT0. IT = 0 – переривання за рівнем (низьким), IT = 1 – переривання за фронтом (перехід із "1" в "0")	Біти встановлюються і скидаються програмно. Доступні за читанням.
3 1	IE1 IE0	Прапорці запиту зовнішніх переривань за входами INT1, INT0.	Біти скидаються і встановлюються апаратно і програмно. Доступні за читанням.
Біти 4, 5 відносяться до Т/Л0; біти 6, 7 – до Т/Л1. *Біти 0, 1 визначають зовнішні переривання за входом INT0, біти 2, 3 – за входом INT1.*			

Прапорці переповнення TF0 і TF1 встановлюються апаратно при переповненні відповідних Т/Л (перехід Т/Л із стану "всі одиниці" у стан "усі нулі").

Якщо при цьому переривання від відповідного Т/Л дозволено, то встановлення прапорця TF викликає переривання. Прапорці TF0 і TF1 скидаються апаратно при передачі керування програмі обробки відповідного переривання.

Прапорці TF0 і TF1 програмно доступні і можуть бути встановлені/скинуті програмою. Використовуючи цей механізм, переривання за TF0 і TF1 можуть бути викликані (встановлення TF) і скасовані (скидання TF) програмою.

Прапорці IE0 і IE1 встановлюються апаратно від зовнішніх переривань (відповідно входи МК INT0 і INT1) або програмно й ініціюють виклик підпрограми обробки відповідного переривання. Скидання цих прапорців виконується апаратно при обслуговуванні переривання тільки в тому випадку, коли переривання було викликано за фронтом сигналу. Якщо переривання було викликано рівнем сигналу на вході INT0 (INT1), то скидання прапорця IE повинна виконувати підпрограма обслуговування переривання, яка повинна вплинути на джерело переривання для зняття ним запиту.

Схему інкременту призначено:

– для збільшення на 1 у кожному машинному циклі вмісту регістрів Т/Л0, Т/Л1, для яких встановлено режим таймера і дозволено лічбу;

– для збільшення на 1 вмісту регістрів Т/Л0, Т/Л1, для яких встановлено режим лічильника, дозволену лічбу і на відповідному вході МК (Т0 для Т/Л0 і Т1 для Т/Л1) зафіксовано лічильний імпульс.

Схема фіксації INT0, INT1, T0, T1 являє собою чотири тригери. У кожному машинному циклі в момент S5 P2 у них запам'ятовується інформація з виводів МК INT0, INT1, T0, T1.

Схема керування прапорцями встановлює і знімає прапорці переповнення Т/Л і прапорці запитів зовнішніх переривань.

Логіка керування Т/Л синхронізує роботу регістрів Т/Л0 і Т/Л1 відповідно до запрограмованих режимів роботи і синхронізує роботу блока Т/Л з роботою МК.

Більш докладно режими роботи й особливості застосування таймерів/лічильників розглянуто у [4…6].

2.8 Блок послідовного порту (інтерфейсу)

Блок послідовного інтерфейсу призначено для організації введення/виведення послідовних даних.

До складу блоку входять: буфер інтерфейсу, логіка керування інтерфейсом, регістр керування, буфер передавача, буфер приймача, приймач–передавач послідовного порту.

Буфер інтерфейсу забезпечує побайтовий обмін інформацією між внутрішньою (резидентною) шиною даних і шиною інтерфейсу.

Логіку керування інтерфейсом призначено для формування сигналів керування, що забезпечують чотири режими роботи послідовного інтерфейсу.

Регістр керування (SCON) призначено для прийому і зберігання коду восьмибітного слова, яке керує послідовним інтерфейсом. Позначення розрядів регістра SCON наведено в таблиці 2.10. Всі розряди регістра SCON програмно доступні для запису ("лог. 0" і "лог. 1") і читання.

Таблиця 2.10 – Позначення розрядів регістра SCON

Біти	7	6	5	4	3	2	1	0
Позначення	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Розряди SM0, SM1 визначають режим роботи інтерфейсу, як зазначено в таблиці 2.11.

Таблиця 2.11 – Вплив розрядів SM0, SM1 регістра SCON на режим роботи інтерфейсу

SM0	SM1	Режим	Найменування	Швидкість передачі
0	0	0	Регістр зсуву	$f_{BQ}/12$
0	1	1	8–бітовий універсальний асинхронний приймач/ передавач (УАПП)	змінна, задається Т/Л1
1	0	2	9–бітовий УАПП	$f_{BQ}/64$ або $f_{BQ}/32$
1	1	3	9–бітовий УАПП	змінна, задається Т/Л1

Інші біти регістра мають таке призначення:

- *SM2* – дозвіл багатопроцесорної роботи. У режимах 2 і 3 при SM2 = 1 прапорець RI не активується, якщо дев'ятий прийнятий біт даних дорівнює "0". У режимі 1 при SM2 = 1 прапорець RI не активується, якщо прийнятий стоп–біт не дорівнює "1". В режимі 0 біт SM2 повинен бути скинутий у "0";
- *REN* – дозвіл прийому послідовних даних. Встановлюється і скидається програмою відповідно для дозволу і заборони прийому;
- *TB8* – дев'ятий біт даних, що передаються, у режимах 2 і 3. Встановлюється і скидається програмою;

– *RB8* – дев'ятий біт прийнятих даних у режимах 2 і 3. У режимі 1, якщо SM2 = 0, RB8 є прийнятим стоп–бітом. У режимі 0 біт RB8 не використовується;
– *TI* – прапорець переривання передавача. Встановлюється апаратно наприкінці часу видачі 8–го біта в режимі 0 або на початку стоп–біта в інших режимах. Скидається програмно;
– *RI* – прапорець переривання приймача. Встановлюється апаратно наприкінці часу прийому 8–го біта в режимі 0 або через половину інтервалу стоп–біта в режимах 1, 2, 3 при SM2 = 0. При SM2 = 1 див. опис для біта SM2;

Буфер передавача призначено для прийому з шини інтерфейсу паралельних даних і видачі їх на передавач послідовного порту.

Буфер приймача служить для прийому даних у паралельній формі від приймача послідовного інтерфейсу.

Буфер приймача і буфер передавача при програмному доступі мають однакове ім'я (SBUF) і адресу (99H). Якщо команда використовує SBUF як регістр джерела, то звернення відбувається до буфера приймача. Якщо команда використовує SBUF як регістр призначення, то звернення відбувається до буфера передавача.

В усіх режимах роботи послідовного порту передача ініціюється будь–якою командою, що використовує SBUF як регістр призначення.

Приймач/передавач послідовного порту призначено для прийому послідовного потоку символів зі входу послідовного порту, виділення даних і видачі їх у буфер приймача, а також для прийому паралельних даних із буфера передавача, перетворення їх у послідовний потік символів і видачі його на вихід послідовного порту.

Більш докладно режими роботи й особливості застосування послідовного інтерфейсу розглянуто у [4…6].

2.9 Паралельні порти введення/виведення

МК AT89C51 містить 4 паралельних 8–розрядних порти введення/ виведення дискретної інформації, що програмуються: P0, P1, P2, P3.

Порти P0, P1, P2, P3 є двонаправленими портами введення/виведення і призначені для забезпечення обміну інформацією МК із зовнішніми пристроями, створюючи 32 лінії введення/виведення. Кожен з портів містить фіксатор–защіпку, що являє собою восьмирозрядний регістр, який має байтову і бітову адресацію для встановлення/скидання його розрядів за допомогою відповідних команд.

Фізичні адреси фіксаторів P0, P1, P2, P3 становлять для:

- P0: 80H, при бітовій адресації: 80H – 87H;
- P1: 90H, при бітовій адресації: 90H – 97H;
- P2: A0H, при бітовій адресації: A0H – A7H;
- P3: B0H, при бітовій адресації: B0H – B7H.

Крім роботи в якості звичайних портів введення/виведення лінії портів P0–P3 можуть виконувати ряд додаткових функцій, які описано нижче.

Через порт P0:

– виводиться молодший байт адреси A0–A7 при роботі з зовнішньою пам'яттю програм і зовнішньою пам'яттю даних;

– видається з МК і приймається в МК байт даних при роботі з зовнішньою пам'яттю (при цьому обмін байтом даних і виведення молодшого байта адреси зовнішньої пам'яті мультиплексовано в часі);

– задаються дані при програмуванні внутрішнього ПЗП, і читається вміст внутрішньої пам'яті програм.

Через порт P1:

– задається молодший байт адреси при програмуванні внутрішнього

ПЗП і при читанні внутрішньої пам'яті програм.

Через порт P2:

– виводиться старший байт адреси A8–A15 при роботі з зовнішньою пам'яттю програм і зовнішньою пам'яттю даних (для зовнішньої пам'яті даних – тільки при використанні команд MOVX A, @DPTR і MOVX @DPTR, A, що формують 16–розрядну адресу);

– задаються старші розряди A8–A11 адреси при програмуванні внутрішнього ПЗП і при читанні внутрішньої пам'яті програм.

Кожна лінія порту P3 має індивідуальну альтернативну функцію:

P3.0 – RxD, вхід послідовного порту, якій призначено для введення послідовних даних у приймач послідовного порту;

P3.1 – TxD, вихід послідовного порту, якій призначено для виведення послідовних даних із передавача послідовного порту;

P3.2 – $\overline{\text{INT0}}$, використовується як вхід 0 зовнішнього запиту переривання;

P3.3 – $\overline{\text{INT1}}$, використовується як вхід 1 зовнішнього запиту переривання;

P3.4 – T0, використовується як вхід лічильника зовнішніх подій Т/Л0;

P3.5 – T1, використовується як вхід лічильника зовнішніх подій Т/Л1;

P3.6 – $\overline{\text{WR}}$, строб запису у зовнішню пам'ять даних, вихідний сигнал, який супроводжує виведення даних через порт P0 при використанні команд MOVX @Ri, A і MOVX @DPTR, A;

P3.7 – $\overline{\text{RD}}$, строб читання із зовнішньої пам'яті даних, вихідний сигнал, який супроводжує введення даних через порт P0 при використанні команд MOVX A, @Ri і MOVX A, @DPTR.

Альтернативна функція будь–якої з ліній порту P3 реалізується тільки в тому випадку, якщо у відповідному цій лінії фіксаторі–защіпці міститься сигнал "лог. 1". Інакше на лінії порту P3 буде присутній сигнал "лог. 0".

2.10 Схема десяткової корекції акумулятора

АЛП МК дозволяє виконувати додавання двійково–десяткових даних в упакованому форматі (в 1 байт “упаковано” 2 десяткові цифри). При виконанні операції додавання таких чисел використовується звичайна команда “Додавання”, що додає операнди за правилами двійкової арифметики і вміщує результат в акумулятор. Для виправлення можливої помилки (корекції вмісту акумулятора) застосовують команду десяткової корекції DA A, яка апаратно реалізується схемою десяткової корекції акумулятора.

2.11 Внутрішній тактовий генератор (OSC)

МК містить внутрішній тактовий генератор (рисунок 2.5), в якому BQ1 і BQ2 є відповідно входом і виходом підсилювача–інвертора, і який може бути ввімкнено у режим генератора при підключенні до виводів BQ1 і BQ2 кварцового резонатора або LC–ланцюжка (рисунок 2.6).

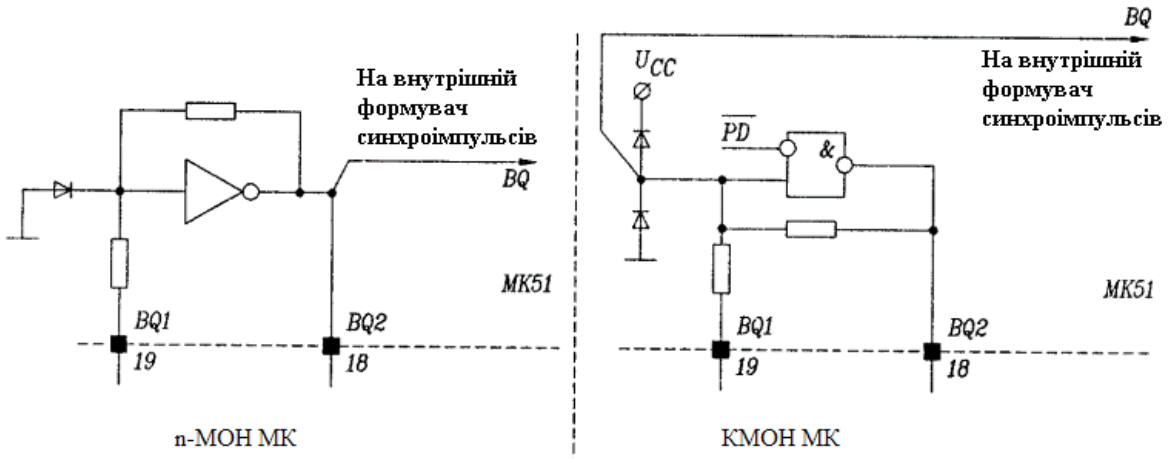


Рисунок 2.5 – Внутрішній тактовий генератор

2.12 Резидентна шина даних

Мікроконтролер містить 8–розрядну внутрішню (резидентну) шину даних (РШД), через яку здійснюється обмін інформацією між різними частинами МК.

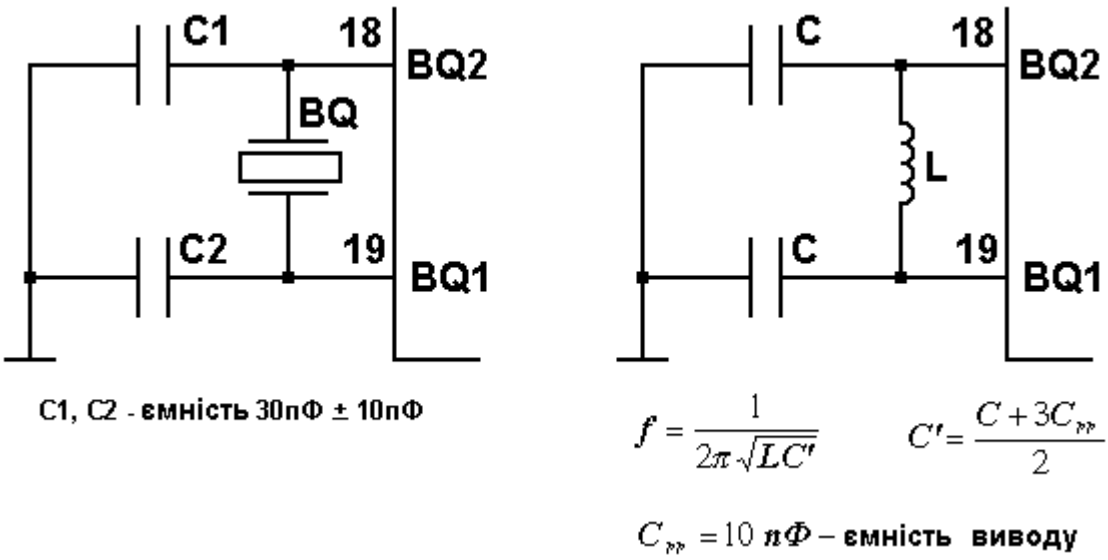


Рисунок 2.6 – Підключення до виводів BQ1 і BQ2 резонатора і LC–ланцюжка

2.13 Регістри

Акумулятор – 8–розрядний регістр, який призначено для прийому і зберігання результату, отриманого при виконанні арифметичних і логічних операцій або операцій пересилання.

Регістр B – 8–розрядний регістр, який використовується при виконанні операцій множення і ділення. В інших випадках він може розглядатися як додатковий регістр загального призначення (частина НОЗП).

Регістри T1, T2 – 8–розрядні регістри тимчасового зберігання, які призначено для прийому і зберігання операндів на час виконання операцій над ними в АЛП. Програмно недоступні.

Регістр стану програми (PSW) служить для зберігання інформації про результат виконання операції в АЛП. Його називають також регістром прапорців (ознак). Нижче наведено призначення окремих розрядів регістра PSW.

Прапорець перенесення CY може встановлюватися і скидатися як апаратно, так і програмно. Апаратно він встановлюється, якщо при виконанні арифметичних або логічних операцій формується перенесення/позика у старшому (сьомому) розряді 8–бітних операндів. При виконанні операцій множення і ділення прапорець CY скидається. Крім того, прапорець CY виконує функції "булевого акумулятора" у командах, що працюють з бітами.

Прапорець допоміжного перенесення AC встановлюється/скидається апаратно або програмно. Апаратно встановлюється при виконанні операцій додавання і віднімання при виникненні перенесення/позики в 3–му розряді при утворенні молодшої тетради результату. Частіше всього використовується схемою СДКА при виконанні команди DA A.

Прапорець користувача F0 встановлюється/скидається програмно і може використовуватися програмістом на свій розсуд.

Прапорці–покажчики поточного банку РЗП встановлюються/ скидаються програмно і вказують, який із 4–х банків РЗП РПД (рисунок 2.4) в даний момент часу є робочим (поточним).

Прапорець переповнення OV встановлюється/скидається програмно або апаратно. Апаратно встановлюється тоді, коли при виконанні операції додавання/віднімання над числами зі знаком результат не вкладається в діапазон –128 ... +127 і старший, знаковий біт спотворюється. При

виконанні операції ділення прапорець OV апаратно скидається, а у випадку ділення на нуль встановлюється. При виконанні операції множення прапорець OV апаратно встановлюється, якщо результат більше 255.

Прапорець парності (паритету) P встановлюється/скидається апаратно. Він доповнює вміст акумулятора до парного числа одиниць. У 9–розрядному слові, що складається з 8 розрядів акумулятора і біта P, завжди міститься парне число одиничних бітів.

Всі сім названих прапорців програмно доступні для читання.

Регістр команд РК (IR) призначено для зберігання коду операції (КОП) поточної команди, що виконується. Програмно не доступний.

Лічильник команд ЛК (PC) містить 16–розрядну адресу комірки пам'яті програм. До складу лічильника команд входять 16–розрядні буфер PC, регістр PC, схема інкременту та регістр адреси пам'яті.

Буфер PC здійснює зв'язок між 8–розрядною РШД і 16–розрядним регістром PC, у якому зберігається поточна 16–розрядна адреса пам'яті програм.

Схема інкременту збільшує поточне значення 16–розрядної адреси пам'яті програм на одиницю.

Регістр адреси пам'яті призначено для запису і зберігання виконавчої 16–розрядної адреси пам'яті програм або 8/16–розрядної адреси зовнішньої пам'яті даних.

Регістр–покажчик даних РГПД (DPTR) призначено для зберігання 16–розрядної адреси зовнішньої пам'яті даних. Складається з двох 8–розрядних регістрів DPH і DPL, що входять у блок регістрів спеціальних функцій [4…6]. Вони програмно доступні і можуть використовуватися в якості двох незалежних РЗП, якщо немає необхідності у зберіганні 16–розрядної адреси зовнішньої пам'яті даних.

Покажчик стеку (SP) адресує комірки спеціальної області пам'яті даних (РПД), яка називається стеком. SP адресує “верхівку” стеку – останню комірку стекової пам'яті, у якій записана інформація. Покажчик стеку являє собою 8–розрядний регістр, вміст якого при виконанні команд LCALL, ACALL збільшується на 2. При виконанні команд RET, RETI вміст покажчика стеку зменшується на 2. При виконанні команди PUSH direct вміст SP збільшується на 1, а при виконанні команди POP direct – зменшується на 1.

Регістр адреси РА (RAR) – регістр комірки РШД, що адресується. Програмно не доступний.

Регістри РРТЛ (TMOD) і *РКСТ (TCON)* служать для програмування і керування роботою таймерів/лічильників і системи переривань. Формати, позначення і призначення їхніх окремих розрядів приведено у таблицях 2.6… 2.9.

Регістр РКПП (SCON), буфери ПД і *ПРМ (SBUF)* призначено для програмування і керування роботою послідовного інтерфейсу. Формати, позначення і призначення їх окремих розрядів приведено в таблицях 2.10, 2.11.

Регістри РМП (IE) і *РП (IP)* програмують і керують роботою підсистеми переривань МК. Формати, позначення і призначення РМП і РП наведено в таблицях 2.3, 2.4.

Регістр керування потужністю РКП (PCON) служить для програм–ного керування споживанням енергії від джерела живлення, а також швидкістю передачі по послідовному каналу. Формати, позначення і призначення його окремих розрядів наведено в таблицях 2.12, 2.13.

Більш докладно застосування цього регістра розглянуто у [4…6].

Таблиця 2.12 – Позначення розрядів регістра PCON

Біти	7	6	5	4	3	2	1	0
Позначення	SMOD	–	–	–	GF1	GF0	PD	IDL

Таблиця 2.13 – Призначення розрядів регістра PCON

Біти	Найменування	Призначення бітів	Примітка
7	SMOD	Біт подвоєння швидкості передачі: при встановленні в "1" – швидкість передачі подвоюється	При роботі послідовного порту
6	–	Резервний	
5	–	Резервний	
4	–	Резервний	
3	GF1	Прапорець загального призначення	
2	GF0	Прапорець загального призначення	
1	PD	Біт вмикання режиму мікроспоживання "1" – режим мікроспоживання	Якщо в PD і IDL одночасно записано "1", перевагу має PD
0	IDL	Біт холостого ходу "1" – режим холостого ходу	

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Назвіть основні функціональні вузли МК.
2) Як співвідносяться між собою командний та машинний цикл?
3) Яке призначення пристрою формування часових інтервалів?
4) Опишіть структуру блоку арифметично–логічного пристрою.
5) Охарактеризуйте розподіл резидентної пам’яті даних та резидентної пам’яті програм.
6) Опишіть структуру підсистеми переривань.
7) Назвіть складові блоку таймерів–лічильників та їх призначення.
8) Опишіть призначення та головні компоненти блоку послідовного інтерфейсу.
9) Назвіть призначення та додаткові функції паралельних портів введення/виведення.
10) Назвіть основні регістри МК–51 та коротко охарактеризуйте їх.

ЛІТЕРАТУРА [4, 5, 6, 7]

3 ПРОГРАМНА МОДЕЛЬ МІКРОКОНТРОЛЕРА

3.1 Місце та роль керуючої програми у роботі мікропроцесорної системи

Якщо мікропроцесорна система (МПС) виконана на основі мікроконтролера (МК), то керуюча програма знаходиться у резидентній (РПП) і/або зовнішній пам'яті програм (ЗПП). Без керуючої програми жодна МПС працювати не буде.

3.2 Послідовність розробки робочої керуючої програми

Основні етапи створення керуючої програми:

– розробка схеми алгоритму, яку повинна реалізувати конкретна керуюча програма;

– якщо задача складна, то програма може бути реалізована і налагоджена на мові високого рівня або мові асемблера;

– компіляція – переведення програми з мови високого рівня або асемблера в машинні коди конкретного МК–ра (створення об'єктного модуля);

– введення керуючої програми в РПП і/або ЗПП за допомогою програматора;

– в процесі початкової ініціалізації системи у програмний лічильник завантажується початкова адреса програми.

Керуюча програма може включати підпрограми.

Підпрограма – це частина основної програми, яка може бути викликана з основної програми або за перериванням.

3.3 Мова асемблера

3.3.1 Загальна характеристика

Основу асемблера складають команди МК–ра, які не можна змінити, а можна вивчити і використовувати під час програмування. Тому говорять, що мову асемблера орієнтовано на конкретний МК–р (на його систему команд).

Окрім цих команд, які при компіляції перетворюються у машинний код, програма включає псевдокоманди та директиви, які вказують програмі компілятора, яка також зветься асемблером, як виконати відповідні дії по створенню файлу для МК–ра, який повинен виконуватися.

Нагадаємо, що при створенні робочих програм для МК використовують такі системні керуючі програми:

– *транслятори* – перетворюють програму, яку написано на одній мові програмування, в програму, яку написано на іншій мові програмування;

– *компілятори* – є різновидом транслятора: переводять програму з мови високого рівня в машинні коди;

– *асемблери* – це окремий випадок компілятора: перетворюють програму з мови асемблера в двійкові (машинні) коди (ДК);

– *налагоджувані* – керуючі програми для налагодження програми у відповідній системі.

3.3.2 Особливості мови асемблера

3.3.3 Структура команди

Кожна команда представляє собою рядок наступної конструкції [3]:

[МІТКА:] мнемокод операції операнд(и) [; коментарі]

[] – Поле може бути відсутнім.

МІТКА – символічне ім'я комірки пам'яті, починаючи з якої дана команда розміщується в пам'яті.

У якості операндів можуть використовуватися числа (адреси і дані), зарезервовані і певні символічні імена.

Для вказівки системи числення, в якій задається число, використовують буквені індекси після самого числа: B – двійкова, Q – вісімкова, D або нічого – десяткова, H – шістнадцяткова.

КОМЕНТАРІ – будь–які символи.

3.3.3.1 Поняття про асемблер (компілятор)

Асемблер – це програмний засіб, призначений для перетворення вихідного тексту програми, що містить мнемонічні імена команд та операндів, в послідовність двійкових кодів, які представляють собою команди для процесора, які повинні виконуватися.

Таким чином, вхідна інформація для асемблера представляється у вигляді текстового файлу, а вихідна інформація генерується у вигляді так званого файлу об'єктних кодів (об'єктного файлу). Крім цього, асемблер виконує перевірку коректного написання команд. Іноді його називають компілятором (транслятором) з мови асемблер.

У результаті роботи асемблера є можливість крім об'єктного файлу отримати також файл лістингу, який містить текстову інформацію про розміщення кодів команд і даних за конкретними адресами пам'яті мікроконтролера або мікропроцесорної системи.

Програма на мові асемблера складається з логічних сегментів – блоків елементів одного типу (команди, дані). Для МК51 ці логічні сегменти безпосередньо відповідають фізичним областям пам'яті (РПП, ЗПП, РПД, ЗПД, бітова область).

Для завдання адрес команд і даних або значень даних зручно використовувати символічні імена (далі – просто імена), які відповідають фізичному або математичному змісту задачі. Використання імен робить програму більш зрозумілою для програміста і його колег, полегшує процес модифікації програми і її налагодження. Ім'я має починатися з літери та містити не більше 32 символів.

Для сегментної побудови програми та визначення символічних імен адрес і значень даних використовують директиви асемблера. Директиви асемблера не є виконуваними командами, а являють собою інструкції для компілятора з розміщення команд і даних у пам'яті МК або МПС.

Структура програми–програма на мові асемблер містить такі блоки:

– декларативна частина – опис символічних імен даних і адрес, які використовують у програмі, а також директиви виділення пам'яті для змінних і завдання значень констант (за допомогою директив асемблера);

– блок ініціалізації – настроювання портів і блоків периферійних функцій на необхідні режими роботи, ініціалізація стеку (за допомогою команд);

– блок реалізації алгоритмів і функцій керування (за допомогою команд).

3.3.3.2 Директиви визначення символічних імен

Директива EQU – будь–якому імені ставиться у відповідність операнд.

Формат директиви: <ім'я> EQU <вираз>

Наприклад:

```
z1 equ 10;
z2 equ z1+4;
z3 equ z1+z2+5.
```

Імена, які визначено директивою EQU, можна використовувати як адресу коду, адресу даних (внутрішніх або зовнішніх) або значення даних. Наприклад:

```
MOV A, # z1; Завантаження числа 10 в акумулятор;
MOV A, z1; Пересилання в акумулятор з пам’яті з адресою 0AH.
```

Оператори, які допустимі для використання у виразах в директивах асемблера, описані далі.

Директива DATA – задає ім'я для адреси даних в РПД.

Формат директиви: <ім'я> DATA <адресний вираз>.

Директива XDATA – задає ім'я для адреси зовнішніх даних (ЗПД).

Формат директиви: <ім'я> XDATA <адресний вираз>.

Директива BIT – задає ім'я для адреси біта з області внутрішнього ОЗП з бітовою адресацією.

Формат директиви: <ім'я> BIT <адреса біта>.

Наприклад:

```
control DATA 2AH;
f1 bit control.3; Адреса у вигляді бітового селектора;
f2 bit f1 4; Адресний вираз;
f3 bit 60H; Абсолютна адреса.
```

Приклади зручного використання символічних імен:

```
counter DATA 20H flag BIT P1.7;
. . . on BIT 30H;
MOV counter, 10. . .;
m1: <дії в циклі> SETB on;
. . . wait: JNB flag, wait;
DJNZ counter, m1 CLR on.
```

3.3.3.3 Лічильник адрес

У процесі компіляції асемблер з кожним сегментом пов'язує свою внутрішню змінну, яка зветься лічильником адрес. У цій змінній підраховується, скільки байтів відведено в пам'яті під кожну команду або число і, відповідно, за якою адресою буде розміщено наступну команду або число. Таким чином, лічильник адрес «стежить» за розміщенням у пам'яті кодів команд і даних.

Позначка команди – це, по суті, символічне ім'я адреси комірки пам'яті, починаючи з якої дану команду розміщено в пам'яті.

Мітки команд зв'язуються з адресами автоматично в процесі компіляції і спеціальних директив для цього не існує.

3.3.3.4 Директиви керування сегментами програми

Поява в тексті програми цих директив означає, що розміщені далі команди або дані відносяться до сегмента (типу) пам'яті, яка зазначена директивою:

CSEG – початок сегмента кодів (пам'ять програм);

DSEG – початок сегмента даних в ОЗП (РПД);

XSEG – початок сегмента зовнішніх даних (ЗПД).

3.3.3.5 Керування значенням лічильника адреси

При першій появі в тексті програми будь–якої з директив керування сегментами асемблер створює новий лічильник адрес для цього сегмента і встановлює його в нуль. При повторному відкритті сегмента продовжується рахунок від попереднього значення.

Існує директива ORG, яка дозволяє програмісту встановлювати потрібні значення лічильника адреси для поточного активного сегмента і таким чином розміщувати коди в пам'яті за потрібними адресами.

Формат директиви: ORG <адресний вираз>.

Директиви ініціалізації пам'яті DB, DW заповнюють байт або слово (2 байти) вказаним значенням. Використовують для розміщення значень констант у сегменті кодів (тобто в пам'яті програм):

[Ім'я:] DB <значення> [ім'я:] DW <значення>.

Директива резервування (виділення) пам'яті DS для змінної в сегменті зовнішніх або внутрішніх даних (ЗПД або ОЗП (РПД)):

[Ім'я:] DS <кількість байтів>.

END – директива визначення кінця програмного модуля.

3.3.3.6 Оператори періоду трансляції

Для зручності запису команд і директив у мові асемблер допускається задавати числові значення та адреси пам'яті у вигляді виразів, які обчислюються. Значення такого виразу буде визначено компілятором ще під час трансляції, а в код команди буде підставлятися тільки результат цих дій. Припустимими операціями є: +, –, *, /, (), AND, OR, XOR, NOT, LOW – виділення молодшого байта, HIGH – виділення старшого байта з двухбайтового виразу.

Наприклад:

1) `X equ 10`; число X = 10

`Y equ X 2`; число Y = 12

`Z equ (X Y) * 4`; число Z = 88

2) `DSEG`

`org 70H`

```
V1: ds 4; Виділення пам'яті для чотирьохбайтової змінної
MOV A, V1 1; Читання в ACC першого байта змінної V1
3) X equ 3704; Оголошення двухбайтового числа
MOV R0, # LOW (X); Запис молодшого байта X в регістр R0
MOV R1, # HIGH (X); Запис старшого байта X в регістр R1
```

3.4 Програмна модель мікроконтролера

3.4.1 Загальна характеристика

Програмна (програмістська) модель включає ті частини архітектури МК, до яких програміст може отримати доступ за допомогою тієї або іншої команди.

Програмну модель мікроконтролера сімейства МК–51 AT89C51 наведено на рисунках 3.1, 3.2 [4–6].

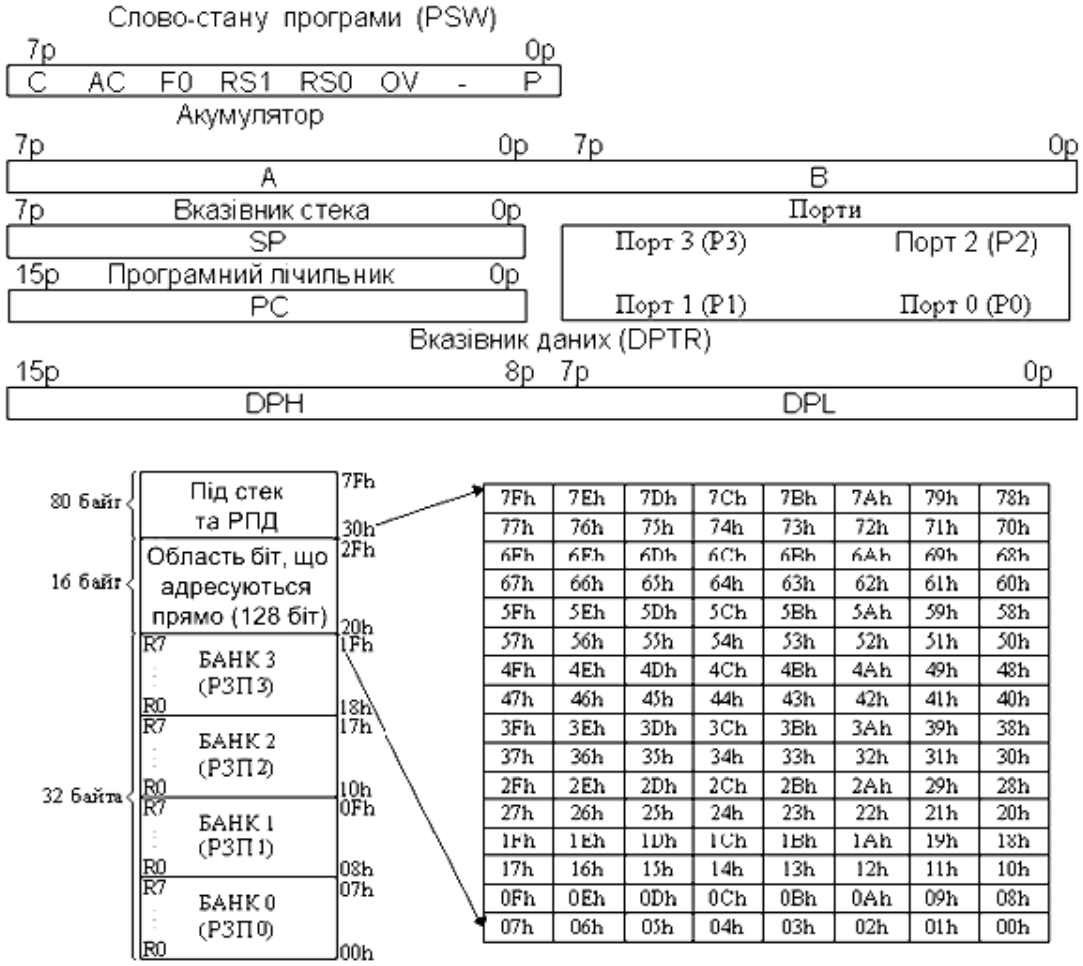
Розглянемо більш детально характеристику окремих складових частин цієї моделі.

3.4.2 Резидентна пам’ять даних

Резидентна пам’ять даних (РПД) призначена для прийому, збереження та видачі інформації, яка використовується в процесі виконання програми. Пам’ять даних ділиться на внутрішню (резидентну) пам’ять даних РПД і зовнішню пам’ять даних ЗПД.

РПД являє собою 128 восьмирозрядних регістрів, які призначені для прийому, збереження та видачі різноманітної оперативної інформації. Шістнадцять із цих регістрів допускають побітову адресацію.

На рисунку 3.3 наведено розподіл адресного простору РПД і області біт, що адресуються прямо.



7Fh	7Eh	7Dh	7Ch	7Bh	7Ah	79h	78h
77h	76h	75h	74h	73h	72h	71h	70h
6Fh	6Eh	6Dh	6Ch	6Bh	6Ah	69h	68h
67h	66h	65h	64h	63h	62h	61h	60h
5Fh	5Eh	5Dh	5Ch	5Bh	5Ah	59h	58h
57h	56h	55h	54h	53h	52h	51h	50h
4Fh	4Eh	4Dh	4Ch	4Bh	4Ah	49h	48h
47h	46h	45h	44h	43h	42h	41h	40h
3Fh	3Eh	3Dh	3Ch	3Bh	3Ah	39h	38h
37h	36h	35h	34h	33h	32h	31h	30h
2Fh	2Eh	2Dh	2Ch	2Bh	2Ah	29h	28h
27h	26h	25h	24h	23h	22h	21h	20h
1Fh	1Eh	1Dh	1Ch	1Bh	1Ah	19h	18h
17h	16h	15h	14h	13h	12h	11h	10h
0Fh	0Eh	0Dh	0Ch	0Bh	0Ah	09h	08h
07h	06h	05h	04h	03h	02h	01h	00h

Адресний простір РПД та область біт, що адресуються прямо та входять до складу РПД

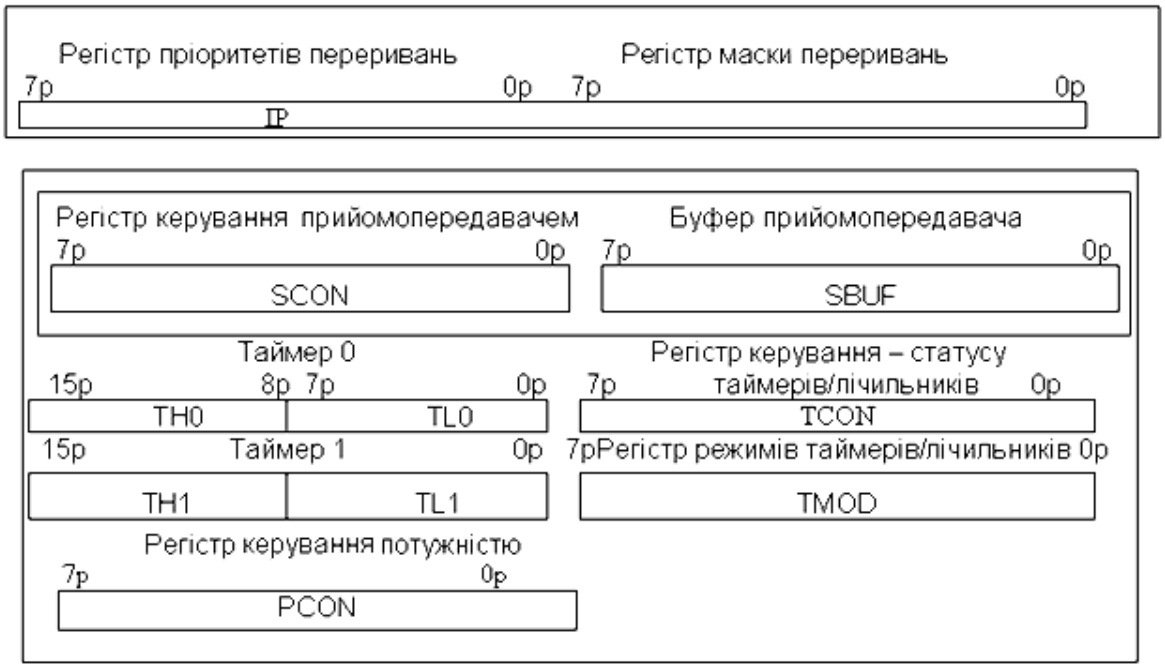


Рисунок 3.1 – Програмна модель МК–51

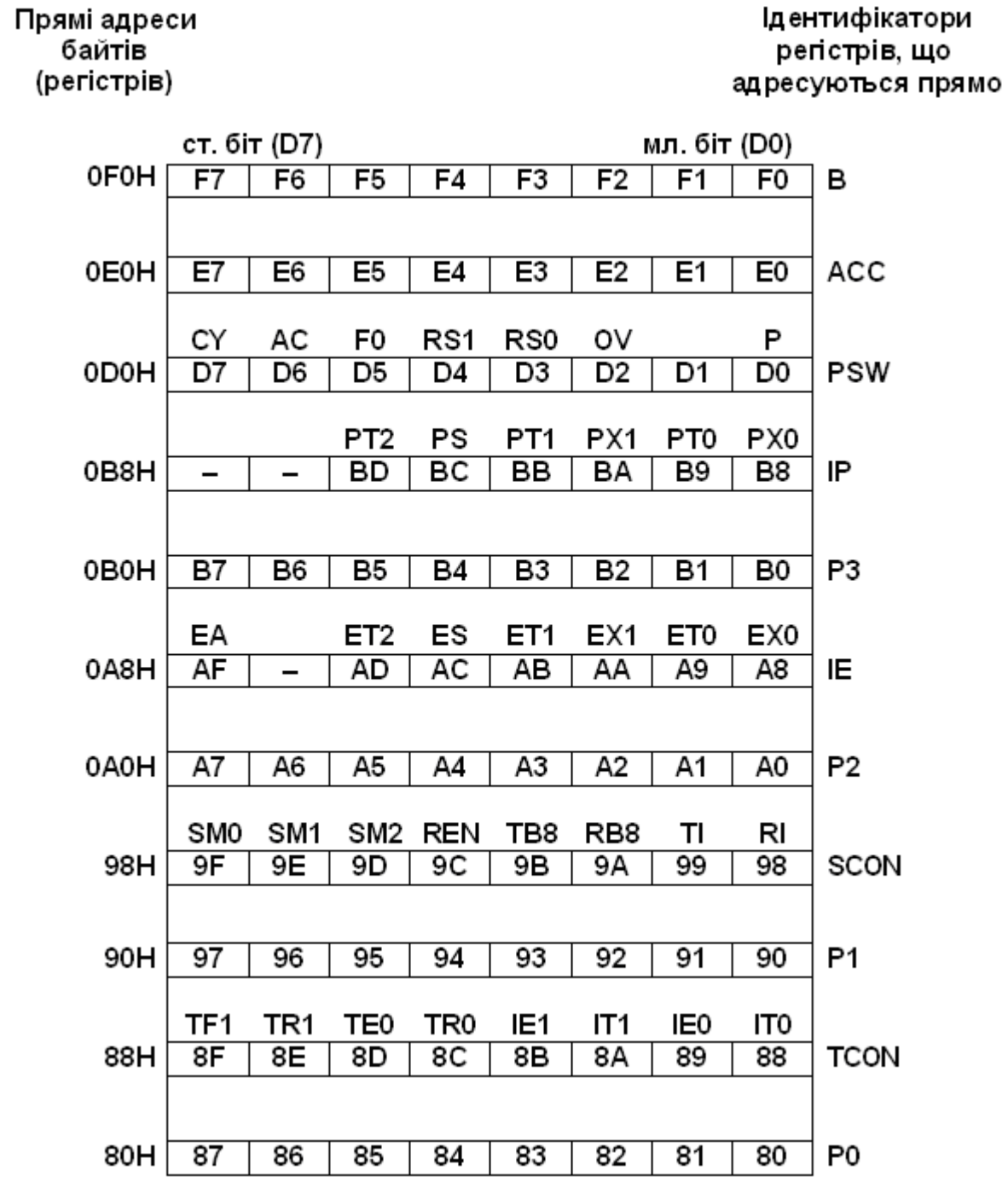


Рисунок 3.2 – Програмна модель МК–51 (продовження)

В області молодших адрес РПД знаходяться 4 банки регістрів загального призначення (РЗП), кожен з яких має об’єм 8 байт (регістрів): R0, R1, …, R7.

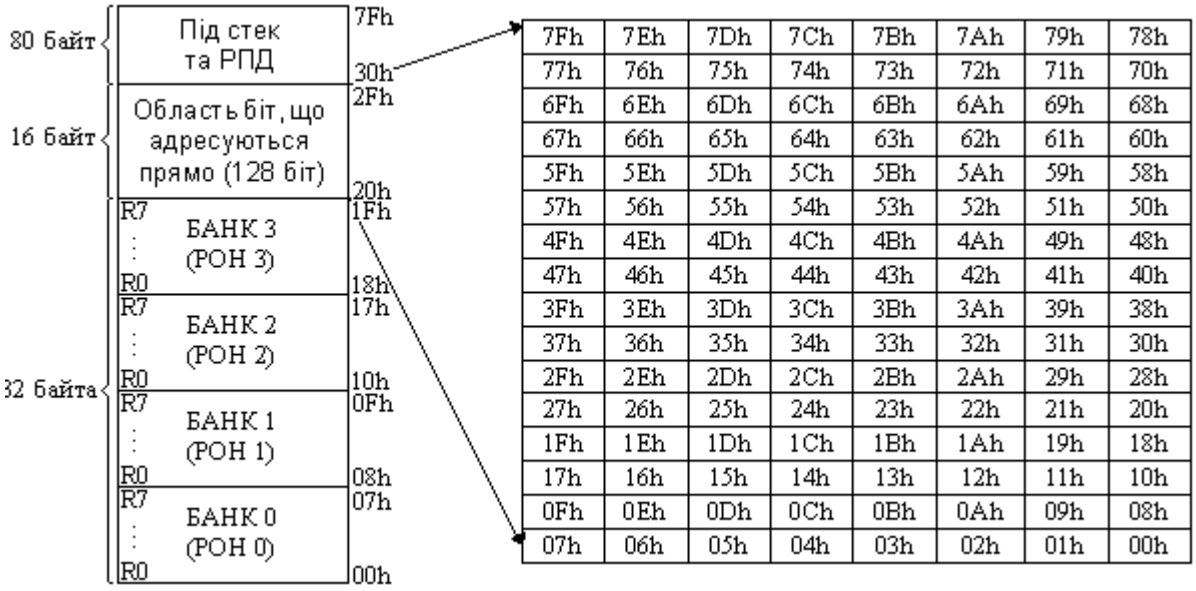


Рисунок 3.3 – Розподіл адресного простору РПД і області біт, які адресуються прямо

3.4.3 Регістри спеціальних функцій

Частину програмної моделі складають регістри спеціальних функцій (таблиця 3.1).

Таблиця 3.1 – Регістри спеціальних функцій

Позначення	Найменування	Адреса
* ACC	Акумулятор	0E0H
* B	Регістр B	0F0H
* PSW	Регістр стану програми	0D0H
SP	Регістр–покажчик стеку	81H
DPTR	Регістр–покажчик даних (2 байти):	
DPL	Молодший байт	82H
DPH	Старший байт	83H
* P0	Порт 0	80H

Продовження таблиці 3.1

Позначення	Найменування	Адреса
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регістр пріоритетів переривань	0B8H
* IE	Регістр дозволу (масок) переривань	0A8H
TMOD	Регістр режимів таймерів/лічильників	89H
* TCON	Регістр керування–статусу таймерів/лічильників	88H
TH0	Таймер/лічильник 0 (старший байт)	8CH
TL0	Таймер/лічильник 0 (молодший байт)	8AH
TH1	Таймер/лічильник 1 (старший байт)	8DH
TL1	Таймер/лічильник 1 (молодший байт)	8BH
* SCON	Регістр керування послідовним портом	98H
SBUF	Буфер послідовного порту	99H
PCON	Регістр керування енергоспоживанням	87H
* – регістри, які допускають побітову адресацію.		

Нижче описано призначення цих регістрів.

Акумулятор A (англ. Accumulator). Один з найважливіших регістрів мікроконтролера. Команди, які призначені для роботи з акумулятором, використовують його ім'я «A», наприклад, MOV A, P2. Ім'я «ACC» використовується, приміром, при побітовій адресації акумулятора. Так, наприклад, символічне ім’я п’ятого біта акумулятора при використанні мови асемблера ASM51 буде таким: ACC.5.

Регістр B. Використовується під час операцій множення та ділення. Для інших команд регістр B може розглядатись як додатковий регістр внутрішнього надоперативного запам`ятовуючого пристрою (НОЗП).

Регістр–покажчик стеку – РПС (англ. Stack Pointer – SP). 8–бітовий регістр, вміст якого інкрементується перед записом даних у стек при виконанні команд PUSH і CALL. При початковому скиданні в покажчик стеку заноситься значення 07H, а область стеку в ОЗП даних починається з адреси 08H. При необхідності, шляхом перевизначення покажчика стеку, область стеку може бути розташована в будь–якому місці внутрішньої пам`яті даних мікроконтролера.

Регістр–покажчик даних (англ. Data Pointer – DPTR). Складається зі старшого байта (DPH) та молодшого байта (DPL). Містить 16–бітову адресу при зверненні до зовнішньої пам'яті. Може використовуватися як 16–бітовий регістр або як два незалежних 8–бітових регістри.

Порт 0 – Порт 3 (англ. Port P0, Port P1, Port P2, Port P3). Виконують функції 32–х ліній введення/виведення, які згруповано в чотири 8–бітові порти.

Буфер послідовного порту (англ. Serial Buffer – SBUF). Фізично являє собою два окремих регістри: буфер передавача та буфер приймача. Коли виконується команда запису даних в SBUF, вони надходять у буфер передавача, причому запис байта в SBUF автоматично ініціює його передачу через послідовний порт. Коли виконується команда зчитування даних з SBUF, то вони вибираються з буфера приймача.

Регістри таймерів (англ. Timer – T). Регістрові пари (TH0, TL0) та (TH1, TL1) утворюють 16–бітові регістри–лічильники відповідно таймер/лічильник0 і таймер/лічильник1.

Регістр стану програми – РСП (англ. Program Status Word – PSW). Регістр PSW містить інформацію про стан програми (таблиця 3.2).

Таблиця 3.2 – Формат регістра РСП

Позиція	Символ	Ім’я та призначення
PSW.7	C	*Прапорець перенесення* (англ. Carry flag). Встановлюється та скидається апаратними засобами при виконанні арифметичних і логічних операцій. Програмно доступний.
PSW.6	AC	*Прапорець допоміжного перенесення* (англ. Auxiliary carry flag). Встановлюється та скидається апаратними засобами при виконанні команд додавання і віднімання та сигналізує про перенесення (переповнення) або позику в біті 3 (вважаючи молодший біт нульовим). Програмно доступний.
PSW.5	F0	*Прапорець F0* (англ. User controlled flag). Може бути встановлений в 0 чи 1 або перевірений програмою як прапорець, який специфікується користувачем.
PSW.4 PSW.3	RS1 RS0	*Вибір банку регістрів загального призначення* (англ. Register select bank switch flag). Встановлюється та скидається програмою для вибору робочого банку регістрів.
PSW.2	OV	*Прапорець переповнення* (англ. Overflow flag). Встановлюється та скидається апаратно при виконанні арифметичних операцій над числами зі знаком. Програмно доступний.
PSW.1	–	*Не використовується*.
PSW.0	P	*Прапорець паритету* (англ. Parity flag). Встановлюється і скидається апаратно в кожному циклі команди та фіксує непарне/парне (1/0) число одиничних біт в акумуляторі, тобто виконує контроль за парністю.

Регістри керування. Регістри спеціальних функцій *IP, IE, TMOD, TCON, SCON і PCON* містять біти керування та біти стану системи переривань, таймерів/лічильників, послідовного порту, схеми керування споживанням енергії від джерела живлення (див. розділ 2):

- регістр пріоритетів переривань РП (англ. Interrupt Pointer – IP);
- регістр дозволу переривань (англ. Interrupt Enable – IE);
- регістр режимів таймерів/лічильників (англ. Timer–Counter Mode – TMOD);
- регістр керування таймерами/лічильниками (англ. Timer–Counter Control – TCON);
- регістр керування послідовним портом (англ. Serial Control – SCON);
- регістр керування енергоспоживанням (англ. Power Control – PCON).

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Назвіть послідовність створення керуючої програми для МК.
2) Визначте призначення мови асемблера.
3) Яким є призначення резидентної пам’яті даних? Як адресується РПД?
4) Перелічіть регістри спеціальних функцій. Які з них допускають пряму бітову адресацію?
5) Опишіть призначення та формат регістра стану програми.
6) Перелічіть та охарактеризуйте регістри керування.
7) Чим відрізняються прапорці C та OV?
8) Чим відрізняються програмістська модель від структури мікроконтролера?
9) Опишіть програмістську модель мікроконтролера AT89C51.

ЛІТЕРАТУРА [2…6]

4 ХАРАКТЕРИСТИКА КОМАНД МІКРОКОНТРОЛЕРА

4.1 Мнемоніка команди та мнемокод

Мнемоніка команди – представлення коду операції у вигляді сполучення латинських літер, що мають визначений зміст (використовуються англійські слова або скорочення, наприклад, MOV, PUSH, POP, JMP, CLR, NOP). Мнемоніки введені для полегшення написання програм і складають основу мови асемблера.

Мнемокод включає в себе мнемоніку команди та опис операндів, які беруть участь в операції.

4.2 Код операції команди

Код операції команди (КОП) – комбінація бітів (не більше восьми для 8–розрядних МК–в), що знаходяться на початку машинного коду команди і визначають тип операції, що підлягає виконанню у даний момент часу. КОП дістається з пам'яті і розміщується в програмно недоступний регістр команд. Потім він декодується і визначається тип команди, яка повинна бути виконана. Для 16–х мікропроцесорів КОП частково може знаходитися у другому байті машинного коду команди. Крім КОП в машинний код команди можуть входити адреси та операнди.

4.3 Машинний код команди

Машинний код команди представляє собою двійковий код команди, який складається з одного або декількох байтів в залежності від типу команди конкретного МК–ра.

4.4 Операнди

Операндами у мікропроцесорній техніці називають дані, які приймають участь у виконанні тієї чи іншої команди (операції). В залежності від способу адресації операндів, дані можуть знаходитися у регістрах, пам’яті, у машинному коді команди і т.ін.

4.5 Коментар

Коментар використовується для полегшення читання програми. Компілятор (асемблер) пропускає коментарі, що показані в програмі через “;” після або до мнемокоду команди, не генеруючи при цьому ніякого машинного коду.

4.6 Формати команд

8–розрядний МК типу MCS–51, наприклад, AT89C51 має команди тринадцяти форматів (типів) (рисунок 4.1). Перший байт команди кожного типу (формату) завжди містить КОП. Другий і третій байти включають в себе або адреси операндів, або безпосередні операнди.

Команди першого типу мають довжину 1 байт. Крім КОП в склад команди може входити адреса одного з восьми РЗП поточного банку регістрів або адреса регістра, що використовувався для непрямої адресації пам'яті.

Команди другого типу містять два байти, з яких:

– перший байт виконує функції, що описані вище;

– другий байт є 8–розрядним безпосереднім операндом (БО), який входить в саму команду.

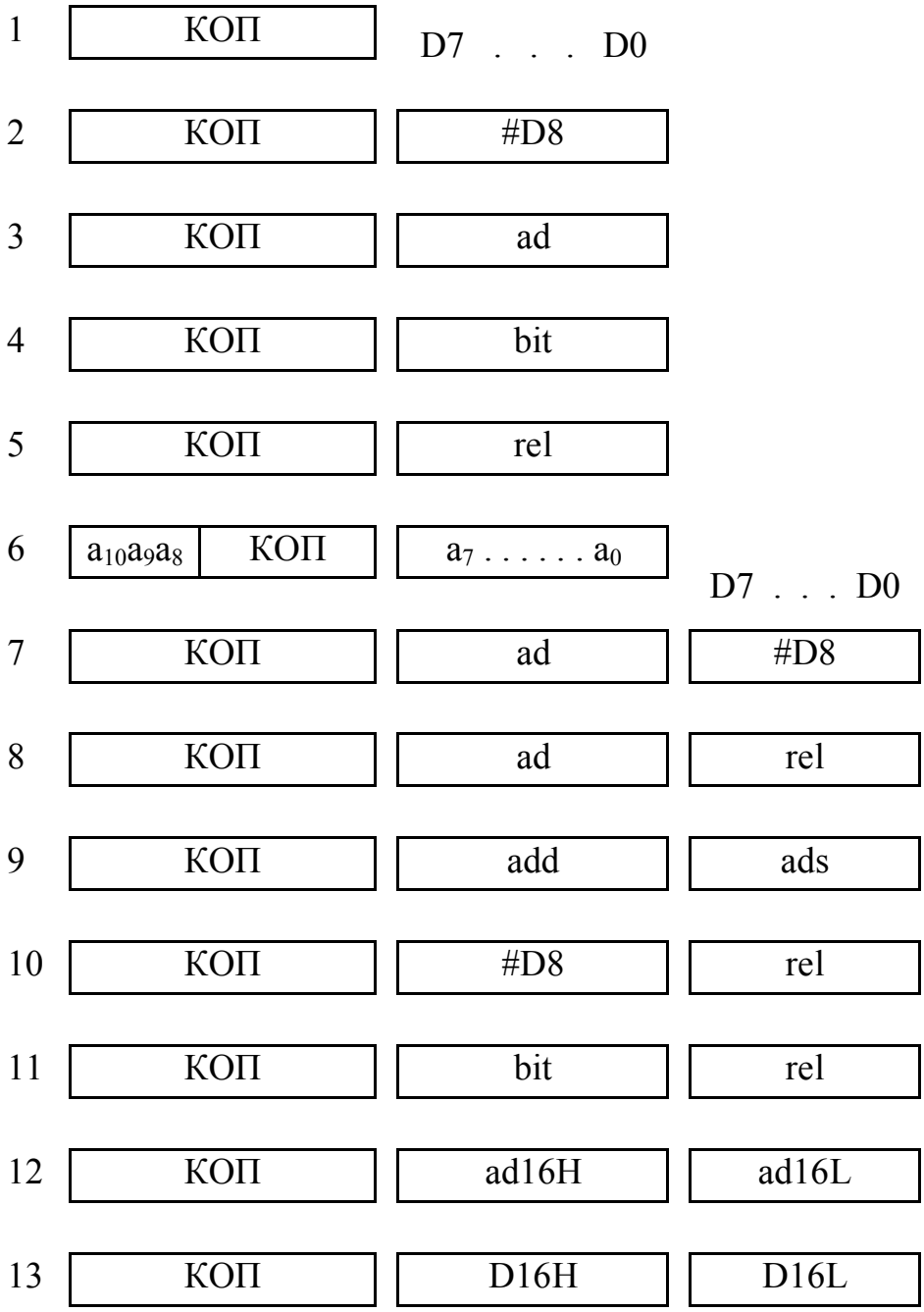


Рисунок 4.1 – Формати команд

Другий байт команд третього типу являє собою пряму адресу комірки резидентної пам’яті даних (РПД) або одного з 21 регістрів спеціальних функцій (таблиця 4.1).

Таблиця 4.1 – Регістри спеціальних функцій

Позначення	Найменування	Адреса
* ACC	Акумулятор	0E0H
* B	Регістр B	0F0H
* PSW	Регістр стану програми	0D0H
SP	Регістр–покажчик стеку	81H
DPTR	Регістр–покажчик даних (2 байти):	
DPL	Молодший байт	82H
DPH	Старший байт	83H
* P0	Порт 0	80H
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регістр пріоритетів переривань	0B8H
* IE	Регістр дозволу (масок) переривань	0A8H
TMOD	Регістр режимів таймерів/лічильників	89H
* TCON	Регістр керування–статусу таймерів/лічильників	88H
TH0	Таймер/лічильник 0 (старший байт)	8CH
TL0	Таймер/лічильник 0 (молодший байт)	8AH
TH1	Таймер/лічильник 1 (старший байт)	8DH
TL1	Таймер/лічильник 1 (молодший байт)	8BH
* SCON	Регістр керування послідовним портом	98H
SBUF	Буфер послідовного порту	99H
PCON	Регістр керування енергоспоживанням	87H
* – регістри, які допускають побітову адресацію.		

Другий байт команд четвертого типу містить адресу одного з 128 біт РПД, які адресуються прямо, (рисунок 4.2) або адресу біт регістрів спеціальних функцій, які допускають пряму бітову адресацію (рисунок 4.3).

Адреса байта	ст. біт (D7)						мол. біт (D0)	
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	42	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	9	8
20H	7	6	5	4	3	2	1	0
1FH	**БАНК 3**							R7
18H								R0
17H	**БАНК 2**							R7
10H								R0
0FH	**БАНК 1**							R7
08H								R0
07H	**БАНК 0**							R7
00H								R0

Рисунок 4.2 – Організація побітової адресації ОЗП

Другий байт команд п'ятого типу є 8–розрядним числом зі знаком в додатковому коді, яке в командах передачі керування додається до поточного значення програмного лічильника (до адреси наступної команди), чим забезпечується перехід в програмі на +127 комірок вперед

або на –128 назад відносно наступної команди. Значення цього числа обчислюється компілятором за формулою:

rel = АДРЕСА МІТКИ – АДРЕСА НАСТУПНОЇ КОМАНДИ (4.1)

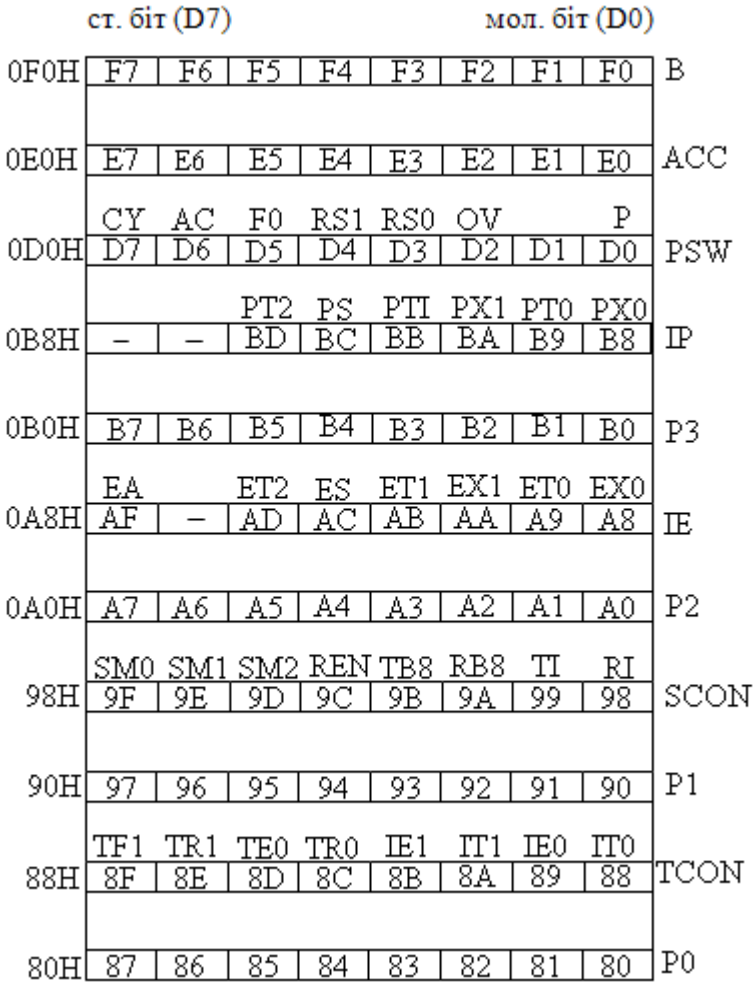


Рисунок 4.3 – Адреси біт регістрів спеціальних функцій

Двобайтова команда шостого типу крім коду операції (КОП) містить одинадцятирозрядну пряму адресу (a10, a9, …, a1, a0), що забезпечує перехід в програмі всередині сторінки, об'ємом 2 Кбайт або виклик підпрограми в тому ж діапазоні адрес.

Другі і треті байти команд 7…11 типів є сполученням описаних вище

складових команд:

– прямої адреси байта (ad, ads, add);

– прямої адреси біта (bit);

– 8–розрядного числа зі знаком (rel);

– безпосереднього операнда (#D8).

Другі і треті байти команд 12–го і 13–го типів представляють відповідно:

– пряму 16–бітову адресу переходу або підпрограми;

– 16–бітовий безпосередній операнд.

4.7 Формати даних

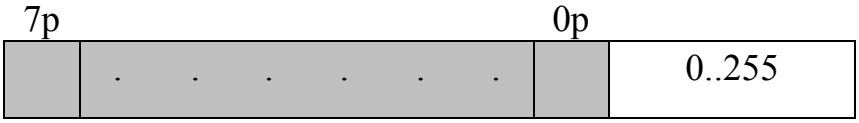
Під час роботи з мікроконтролером необхідно знати не тільки формати команд, які керують роботою МК, але й формати даних (операндів), що приймають участь у виконанні тієї або іншої команди (операції) (рисунок 4.4).

4.8 Довжина команд та їх розміщення у пам'яті програм

У 8–розрядному МК–рі, наприклад, AT89C51, багатобайтові команди зберігаються в сусідніх комірках пам’яті та адресуються за першим байтом, причому двобайтові слова даних і адреси розташовуються в порядку зростання адреси пам`яті – спочатку старший, а потім молодший байти. Як видно з рисунка 4.1, для цього МК–ра команди можуть мати довжину 1,2 або 3 байти.

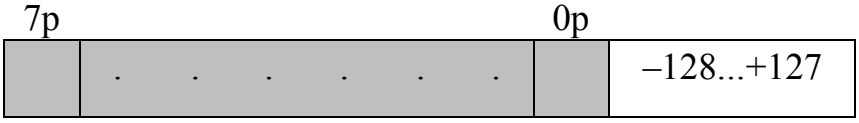
Програма, яка керує роботою МК–ра, послідовно, команда за командою, розміщується в сусідніх комірках пам'яті в порядку зростання їх адрес. Адреса команди, яка повинна використовуватись, знаходиться у програмному лічильнику PC.

Байт без знака:



СЗР, старший значущий розряд — МЗР, молодший значущий розряд — ДДЧ, (діапазон десяткових чисел)

Байт зі знаком:



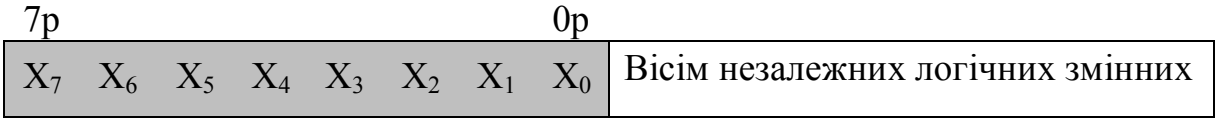
СЗР — МЗР — ДДЧ

знак СЗР:

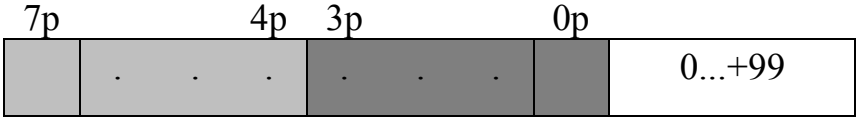
1 – мінус

0 – плюс

Логічний байт:



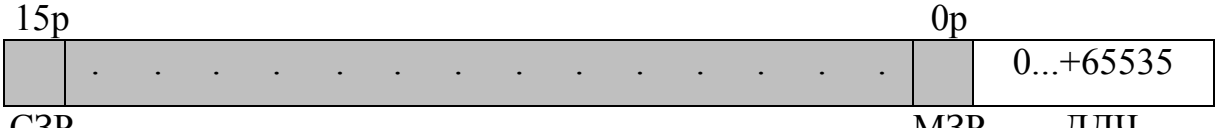
Упаковане двійково–десяткове число:



СЗР — МЗР — ДДЧ

старша тетрада (0…9D) — молодша тетрада (0…9D)

Двобайтове слово:



СЗР — МЗР — ДДЧ

Рисунок 4.4 – Формати (типи) даних

Пристрій керування мікроконтролером на підставі прочитаного коду операції, що міститься завжди в першому байті команди, визначає, скільки ще байтів міститься в команді, та керує їх читанням з пам'яті шляхом збільшення на одиницю адреси, яка видається при кожному зверненні до ЗП. Після читання з пам'яті чергової команди МК формує адресу початку (КОП) наступної команди.

4.9 Вплив команд на прапорці

Як відзначалося раніше, одним з основних регістрів МК–ра є регістр прапорців (ознак).

Прапорці регістра ознак встановлюються під час виконання ряду команд МК–ра. Під час опису системи команд, яка, як правило, оформлюється у вигляді таблиці, в одній з колонок показується вплив окремих команд на ті або інші прапорці [4 – 6].

Якщо команда не змінює прапорець, то ставлять мінус, якщо змінює, то ставлять плюс, а якщо встановлює у конкретний стан, то пишуть нуль (0) або одиницю (1). Як правило, команди пересилок не змінюють прапорці, окрім команд, які пересилають дані у регістр прапорців. Частіше прапорці змінюють арифметичні, логічні команди і команди порівняння.

Окремі прапорці (ознаки) аналізуються під час виконання команд умовних переходів, виклику і повернення з підпрограм, що дозволяє передавати керування в програмі в залежності від поточного значення прапорців.

В таблиці 4.2 показано вплив на прапорці команд МК типу AT89C51.

Примітки:

1. x – прапорець, дорівнює 0 або 1.

2. Операції над вмістом регістра прапорців PSW або його окремих розрядів також впливають на встановлення прапорців.

3. Прапорець паритету (парності) P встановлюється/скидається апаратно та відображає число одиниць в акумуляторі.

Таблиця 4.2 – Команди, що впливають на встановлення прапорців МК

Мнемоніка (мнемокод)	Прапорці		
	C	OV	AC
ADD	x	x	x
ADDC	x	x	x
SUBB	x	x	x
MUL	0	x	
DIV	0	x	
DAA	x		
RRC	x		
RLC	x		
SETB C	1		
CLR C	0		
CPL C	x		
ANL C, bit	x		
ANL C, /bit	x		
ORL C, bit	x		
ORL C, /bit	x		
MOV C, bit	x		
CJNE	x		

4.10 Час виконання команд

В МК–рі типу MCS–51 окремі команди виконуються за 1, 2 або 4 машинні цикли.

Всі машинні цикли однакові і складаються з 6 станів: (S1, S2, .., S6)

по дві фази в кожному стані (Р1, Р2). Тривалість однієї фази дорівнює періоду тактової частоти f_{BQ}. Тобто один машинний цикл складається з 12 фаз (12 періодів f_{BQ}). Знаючи значення f_{BQ}, можна визначити $T_{BQ} = 1/f_{BQ}$, а отже, і час машинного циклу $T_{МЦ} = T_{BQ}*12$.

Більшість команд МК типу MCS–51 виконуються за 1 або 2 машинні цикли, а множення і ділення – за 4 машинні цикли. Якщо, наприклад, f_{BQ} = 12 МГц, то $T_{BQ} = 1/12*10^{-6}$ с, $T_{МЦ} = (1/12)*12*10^{-6} = 1$ мкс, а окремі команди виконуються за 1, 2 або 4 мкс.

4.11 Способи адресації операндів

4.11.1 Загальна характеристика

Тип звернення (адресації) до операндів (даних, що беруть участь в операції) називають *способом адресації*. До способів адресації операндів також відносять те, як в командах передачі керування дається вказівка на адресу переходу.

У МК типу MCS–51 існують наступні способи адресації операндів – даних, що беруть участь в операції:

- неявна;
- регістрова (пряма регістрова);
- пряма (байтова і бітова);
- безпосередня;
- непряма;
- відносна;
- базово–індексна;
- стекова.

При *неявній* адресації інформацію про адресу операнда, який бере участь в операції, пристрій керування МК–ра одержує з коду операції

команди. Таким чином часто адресуються акумулятор і прапорець перенесення, наприклад:

INC A,

CLR C.

При *регістровій* адресації команда містить трьохрозрядну пряму адресу одного з восьми робочих регістрів поточного банку регістрів загального призначення (РЗП) резидентної пам’яті даних (РПД). Для вибору робочого (поточного) банку використовуються розряди PSW3, PSW4 (RS0, RS1) регістра прапорців. Приклади команд з регістровою адресацією:

MOV A, R5;

XCH A, R3.

Пряма байтова адресація застосовується для звернення до комірок РПД і до регістрів спеціальних функцій. В цьому випадку команда включає в себе пряму 8–розрядну адресу операнда, яка при описанні команд мікроконтролера позначається ad, наприклад:

ADD A, ad;

DEC ad.

Пряма бітова адресація використовується для звернення до 128 бітів РПД, які адресуються окремо, і до бітів регістрів спеціальних функцій, які адресуються окремо, (див. програмістську модель МК–ра). При цьому команда містить пряму 8–розрядну адресу біта, який бере участь в операції. Ця адреса при описанні команд позначається bit, наприклад:

SET bit;

MOV C, bit.

При *безпосередній* адресації операнд входить до складу самої команди і дістається з пам'яті при виконанні команди одразу ж після її

коду операції. Безпосередній операнд позначається #D8, #D16, наприклад:

MOV A, #D8;

MOV DPTR, #D16.

При *непрямій* адресації в команді міститься посилання на регістр, в якому міститься адреса операнда. Ця адресація може використовуватися для звернення до комірок РПД або зовнішньої пам'яті даних ЗПД. В якості регістрів–вказівників РПД використовуються регістри R0, R1 робочого (поточного) банку РЗП. Це, наприклад, наступні команди:

MOV A, @Ri;

MOV ad, @Ri.

В якості регістрів–вказівників ЗПД застосовуються ті ж R0 і R1, що дозволяє вибрати одну з 256 комірок зовнішньої пам'яті даних, або 16–розрядний регістр–вказівник даних (DPTR), який забезпечує адресацію однієї з 65536 = 64 К восьмирозрядних комірок ЗПД. Наприклад, наступні команди:

MOVX A, @Ri;

MOVX @DPTR, A.

Відносна адресація застосовується в командах переходів для обчислення адреси команди, на яку передається керуванння. При цьому в команді задається 8–розрядний зсув відносно адреси самої команди, який при описанні команд позначається як rel. Зсув сприймається як число зі знаком, що представлене у додатковому коді. Це дозволяє здійснювати переходи на +127 байт вперед і на –128 байт назад відносно адреси наступної команди. Наприклад, наступні команди:

JNZ rel;

DJNZ ad, rel,

де rel – 8–бітний зсув. Зсув обчислюється компілятором шляхом віднімання від адреси мітки команди, на яку потрібно перейти, адреси наступної команди.

При *базово–індексній* адресації адреса операнда в пам'яті програм обчислюється як сума 16–розрядної базової адреси, що міститься в регістрах DPTR або PC, і 8–бітного індексу, що знаходиться в акумуляторі. Це дозволяє звертатися до елементів таблиць, масивів і т.ін. Наприклад, наступні команди:

MOVC A, @A+DPTR;

MOVC A, @A+PC.

Стекова адресація виділена в самостійний вид адресації умовно, щоб підкреслити специфіку роботи зі спеціальною областю пам'яті, яка називається стеком.

При записі до стеку (команда PUSH) стек розширюється, а при читанні зі стеку (команда POP) стек стискається. В будь–якому випадку операція виконується відносно крайньої (останньої) заповненої комірки, яка називається "вершиною стеку". Формально стек – це пам'ять з організацією типу "останній увійшов – перший вийшов".

У мікроконтролері MCS–51 при виконанні команди PUSH початкове значення SP збільшується на одиницю та за отриманою адресою байт записується у стек. При виконанні команди POP зі стеку зчитується байт за початковою адресою SP, після чого значення SP зменшується на 1.

4.11.2 Приклади команд, які використовують різні способи адресації операндів

1) MOV A, R4; A ← R4.

Пересилання в акумулятор із регістра.

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – регістрова.

2) MOV 10h, @R1; ad=10h← РПД(R1).

Пересилання байта з РПД за прямою адресою байта.

Спосіб адресації операндів :

лівий операнд – пряма байтова;

правий операнд – непряма байтова.

3) MOV DPTR, #0A65Bh; DPTR ← 0A65Bh.

Завантаження регістра–покажчика даних.

Спосіб адресації операндів:

лівий операнд – неявна двобайтова;

правий операнд – безпосередня двобайтова.

4) MOV 3h, 5h; (add) ← (ads).

Пересилання байта, що адресується прямо, за прямою адресою.

Спосіб адресації операндів:

лівий операнд – пряма байтова;

правий операнд – пряма байтова.

5) MOV C, 8h; (C) ← (b).

Пересилання біта, що адресується прямо, у перенесення.

Спосіб адресації операндів :

лівий операнд – неявна бітова;

правий операнд – пряма бітова.

6) MOVX A, @R1; (A) ← ЗПД(R1).

Пересилання в акумулятор байта із ЗПД.

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – непряма.

7) XCH A, 80h; (A) ↔ (ad).

Обмін акумулятора з байтом, що адресується прямо.

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – пряма байтова.

8) ADD A, R3; (A) ← (A) + (R3).

Додавання акумулятора з регістром R3.

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – регістрова.

9) ADDC A, #82h; (A) ← (A) + D8 + (C).

Додавання акумулятора з константою і перенесенням.

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – безпосередня; С–неявна;

10) SUBB A, 15h; (A) ← (A) – (C) – (ad).

Віднімання з акумулятора байта, що адресується прямо, і запозичення (С).

Спосіб адресації операндів :

лівий операнд – неявна байтова;

правий операнд – пряма байтова;

С– неявна.

11) INC @R1; РПД(R1) ← РПД(R1) + 1.

Інкремент байта в РПД.

Спосіб адресації операндів : непряма.

12) MUL AB; якщо (A)x(B) > 255, то (B)(A) ← (A)x(B); якщо (A)x(B) < 255, то (A) ←(A)x(B).

Множення акумулятора на регістр B.

Спосіб адресації операндів : неявна.

13) ANL 0F0h, #0A5h; (0F0h) ← (0F0h) Λ D8=#0A5h.

Команда байтової логічної операції кон'юнкція між регістром спеціальних функцій з адресою 0F0h і безпосереднім числом 0A5h.

Спосіб адресації операндів :

лівий операнд – пряма байтова;

правий операнд – безпосередня.

14) RR A; (An) ← (An+1), n=0..6, (A7) ← (A0).

Команда циклічного зсуву акумулятора вправо.

Спосіб адресації операндів :

Неявна байтова.

15) ORL C, 23h; (C) ← (C) V (23h).

Команда бітової логічної операції диз'юнкція між прапорцем перенесення C і бітом з адресою 23h.

Спосіб адресації операндів :

лівий операнд – неявна бітова;

правий операнд – пряма бітова.

16) SETB TCON.4; (TCON.4) ← 1.

Команда встановлення біта TCON.4.

Спосіб адресації операндів : пряма бітова.

17) SJMP M1; (PCпоч = 4Ch; ADR M1 = 26h); (PC) ← (PC) + 2, (PC) ← (PC) + rel.

Команда переходу на мітку M1.

Спосіб адресації операндів : відносна.

18) JZ M1; (PCпоч = 13h; ADR M1 = 26h); (PC) ← (PC) + 2, якщо A = 0, то (PC) ← (PC) + rel.

Команда умовного переходу. Перехід на мітку M1, якщо акумулятор дорівнює нулю.

Спосіб адресації операндів : відносна.

19) JNC M2; (PCпоч = 7Fh; ADR M2 = 0A8h); (PC) ← (PC) + 2, якщо (C) = 0, то (PC) ← (PC) + rel;

Команда умовного переходу. Перехід на мітку M2 якщо прапорець перенесення дорівнює нулю.

Спосіб адресації операндів : відносна

20) JB 18h, M3; (PCпоч = 2Bh; ADR M3 = 56h); (PC) ← (PC) + 3, якщо (18h) = 1, то (PC) ← (PC) + rel.

Команда умовного переходу. Перехід на мітку M3 якщо встановлено біт 18h.

Спосіб адресації операндів :

лівий операнд – пряма бітова;

правий операнд – відносна.

21) JNB 7h, M1; (PCпоч = 7Dh; ADR M1 = 26h); (PC) ← (PC) + 3, якщо (7h) = 0, то (PC) ← (PC) + rel.

Команда умовного переходу. Перехід на мітку M1, якщо біт 7h дорівнює нулю.

Спосіб адресації операндів :

лівий операнд – пряма бітова;

правий операнд – відносна.

22) JBC 78h, M2; (PCпоч = 62h; ADR M2 = 0A8h); (PC) ← (PC) + 3, якщо (78h)= 1, то (78h) = 0 та (PC) ← (PC) + rel.

Команда умовного переходу з наступним скиданням біта. Перехід на мітку M2, якщо біт з адресою 78h був встановлений, і скидання біта з адресою 78h.

Спосіб адресації операндів :

лівий операнд – пряма бітова;

правий операнд – відносна.

23) DJNZ R3, M3; (PCпоч = 7Bh; ADR M3 = 56h; R3 = 4); (PC) ← (PC) + 2, R3 ← R3 – 1, якщо R3 ≠ 0, то (PC) ← (PC) + rel.

Команда умовного переходу. Декремент регістра R3 і перехід на мітку M3, якщо значення регістра R3 не дорівнює нулю.

24) CJNE A, 13h, M1; (PCпоч = 03h; ADR M1 = 26h); (PC) ← (PC) + 3, якщо A ≠ (13h), то (PC) ← (PC) + rel; якщо A <(13h), то (C) ← 1, або (C) ← 0.

Команда умовного переходу. Перехід на мітку M1, якщо значення акумулятора не дорівнює значенню байта з адресою 13h. У разі, якщо значення акумулятора менше значення байта з адресою 13h, то встановлюється прапорець перенесеня, інакше цей прапорець скидається.

Спосіб адресації операндів :

– перший операнд– акумулятор A – неявна;

– другий операнд – пряма байтова;

– перехід на мітку M1– відносна.

25) CJNE R1, #84h, M2; (PCпоч = 38h; ADR M2 = 0A8h); (PC) ← (PC) + 3, якщо R1 ≠ 84h, то (PC) ← (PC) + rel; якщо R1 < 84h, то (C) ← 1, інакше (C) ← 0.

Команда умовного переходу. Перехід на мітку M2, якщо значення R1 не дорівнює значенню 84h. У разі, якщо значення регістра R1 менше значення 84h, то встановлюється прапорець переносення, інакше цей прапорець скидається.

Спосіб адресації операндів :

перший операнд –регістрова;

другий операнд – безпосередня;

третій операнд – відносна.

4.12 Класифікація команд за функціональним призначенням

4.12.1 Загальні відомості

111 основних команд мікроконтролера MCS–51 розбиті на п'ять груп (таблиці 4.3–4.7) [4–6]:

– команди передачі даних;

– арифметичні команди;

– логічні операції;

– операції з бітами;

– команди передачі керування.

Таблиці команд дають достатньо повну характеристику кожної з базових команд мікроконтролера:

– назва команди (у стислій формі описує функцію команди);

– мнемокод (включає в себе мнемоніку команди та опис операндів, які беруть участь в операції);

– код операції (КОП);

– тип команди (позначається як Т);

– довжину команди в байтах (позначається як Б);

– кількість машинних циклів, необхідних для виконання команди (позначається як Ц);

– коментар (операція).

4.12.2 Символічні позначення, які використовуються при описі команд мікроконтролера

При описі команд мікроконтролера (таблиці 4.3–4.7) використовуються наступні символічні позначення:

1) A – регістр–акумулятор (8 біт);
2) Rn – один з регістрів загального призначення поточного банку РЗП (R0/R1/R2/R3/R4/R5/R6/R7; n = 0, 1, …, 7). RS1, RS0 – два біти регістра прапорців – PSW (рисунок 4.5);

Вибір поточного банку РЗП		
№банку	RS1	RS0
0	0	0
1	0	1
2	1	0
3	1	1

Рисунок 4.5 – Вибір поточного банку РЗП

3) @Ri – непряма адресація. Операнд знаходиться в резидентній пам'яті даних (РПД) або в зовнішній пам'яті даних (ЗПД) за адресою, що міститься в одному з двох РЗП поточного банку: R0/R1 (Ri, де i = 0/1);
4) ad; add; ads – позначення байта, що адресується прямо, який міститься в РПД або в одному з 21 регістрів спеціальних функцій (РСФ);
5) #D8 – 8–розрядний безпосередній операнд (константа);
6) DPTR – 16–розрядний регістр МК–ра;
7) PC – програмний лічильник – 16–розрядний регістр МК–ра;

8) @DPTR – непряма адресація. Операнд знаходиться у зовнішній пам'яті даних (ЗПД) за адресою, що міститься в регістрі DPTR;
9) @A+DPTR; @A+PC – непряма (базово–індексна) адресація. Операнд знаходиться в пам'яті програм (ПП) за адресою, яка обчислюється відповідно як сума: (A+DPTR) або (A+PC);
10) B – регістр–розширювач акумулятора (8 біт);
11) bit – позначення біта, що адресується прямо, який міститься в одному з 128 бітів РПД (8*16 = 128) або в 11 регістрах спеціальних функцій (див. програмістську модель);
12) C – прапорець перенесення/позики регістра PSW;
13) ad16; ad11 – відповідно 16–ти або 11–ти розрядна пряма адреса пам'яті програм;
14) rel – 8–ми розрядне число зі знаком: (–128...+127), яке при виконанні команд переходів додається до адреси наступної команди (rel = ADR мітки – ADR наступної команди);

4.12.3 Команди передачі данних

Більшу частину команд даної групи (таблиця 4.3) складають команди передачі та обміну байтами. Команди передачі (пересилки) входять також в групу команд роботи з окремими бітами. Всі команди даної групи не модифікують прапорці результату, за винятком команд завантаження PSW, тригера (прапорця) C і акумулятора (встановлюється прапорець паритету). На рисунку 4.6 зображено граф шляхів передачі даних в МК–рі.

Окремою вершиною на цьому графі зображено акумулятор A, тому що він бере участь у більшості пересилок і адресується за допомогою неявної або прямої адресації. Проте деякі пересилки виконуються без участі акумулятора.

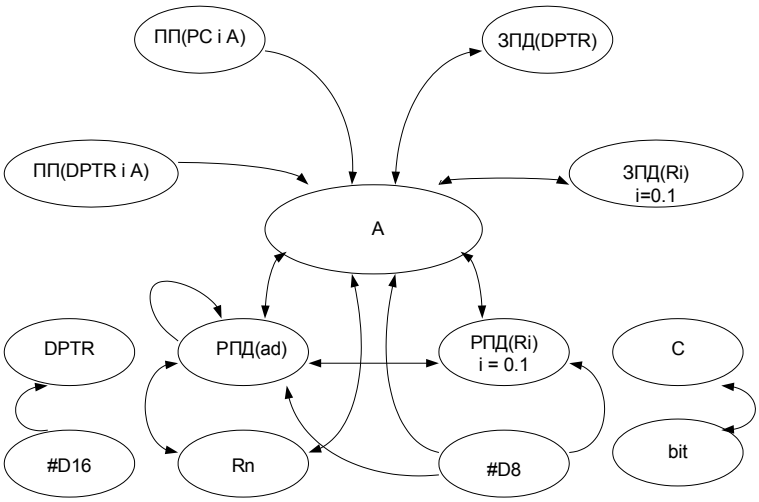


Рисунок 4.6 – Граф шляхів передачі даних в МК–рі

В число команд пересилок входять операції зі стеком, який організується в РПД. Для адресації комірок стеку використовується регістр–покажчик стеку (SP), який дозволяє адресувати будь–яку комірку внутрішнього ОЗП. Запис інформації в стек проводиться командою PUSH ad, а зчитування зі стеку – POP ad. У початковоому стані SP адресує "вершину" стеку – останню комірку стекової пам’яті, у якій записана інформація.

Довжина операнда при роботі зі стеком дорівнює 8 біт. Перед записом у стек вміст SP інкрементується, а після зчитування – декрементується.

Таблиця 4.3 – Група команд передачі даних

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
1	Пересилка в акумулятор з регістра (n = 0..7)	MOV A, Rn	11101rrr	1	1	1	(A) ← (Rn)
2	Пересилка в акумулятор байта, що адресується прямо	MOV A, ad	11100101	3	2	1	(A) ← (ad)
3	Пересилка в акумулятор байта з РПД (i = 0,1)	MOV A, @Ri	1110011i	1	1	1	(A) ← РПД(Ri)
4	Завантаження в акумулятор константи	MOV A, #D8	01110100	2	2	1	(A) ← D8
5	Пересилка у регістр (n = 0..7) з акумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) ← (A)
6	Пересилка у регістр байта, що адресується прямо	MOV Rn, ad	10101rrr	3	2	2	(Rn) ← (ad)
7	Завантаження в регістр (n = 0..7) константи	MOV Rn, #D8	01111rrr	2	2	1	(Rn) ← D8
8	Пересилка за прямою адресою акумулятора	MOV ad, A	11110101	3	2	1	(ad) ← (A)
9	Пересилка за прямою адресою регістра	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
10	Пересилка байта, що адресується прямо, за прямою адресою	MOV add, ads	10000101	9	3	2	(add) ← (ads)
11	Пересилка байта з РПД за прямою адресою	MOV ad, @Ri	1000011i	3	2	2	(ad) ← РПД(Ri)
12	Пересилка за прямою адресою константи	MOV ad, #D8	01110101	7	3	2	(ad) ← D8
13	Пересилка у РПД з акумулятора	MOV @Ri, A	1111011i	1	1	1	РПД(Ri) ← (A)
14	Пересилка у РПД байта, що адресується прямо	MOV @Ri, ad	0110011i	3	2	2	РПД(Ri) ← (ad)
15	Пересилка у РПД константи	MOV @Ri, #D8	0111011i	2	2	1	РПД(Ri) ← D8
16	Завантаження регістра–покажчика даних	MOV DPTR, #D16	10010000	13	3	2	(DPTR) ← D16
17	Пересилка в акумулятор байта з ПП	MOVC A, @A+DPTR	10010011	1	1	2	(A) ←ПП((A) + (DPTR))
18	Пересилка в акумулятор байта з ПП	MOVC A, @A+PC	10000011	1	1	2	(PC) ← (PC) + 1 (A) ← ПП((A) + (PC))
19	Пересилка в акумулятор байта з ЗПД	MOVX A, @Ri	1110001i	1	1	2	(A) ← ЗПД(Ri)
20	Пересилка в акумулятор байта з розширеної ЗПД	MOVX A, @DPTR	11100000	1	1	2	(A) ← ЗПД(DPTR)
21	Пересилка у ЗПД з акумулятора	MOVX @Ri, A	1111001i	1	1	2	ЗПД(Ri) ← (A)
22	Пересилка у розширену ЗПД з акумулятора	MOVX @DPTR, A	11110000	1	1	2	ЗПД(DPTR) ← (A)
23	Завантаження у стек	PUSH ad	11000000	3	2	2	(SP) ← (SP) + 1 РПД(SP) ← (ad)
24	Зчитування зі стеку	POP ad	11010000	3	2	2	(ad) ← РПД(SP) (SP) ← (SP) – 1
25	Обмін акумулятора з регістром	XCH A, Rn	11001rrr	1	1	1	(A) ↔ (Rn)
26	Обмін акумулятора з байтом , що адресується прямо	XCH A, ad	11000101	3	2	1	(A) ↔ (ad)
27	Обмін акумулятора з байтом з РПД	XCH A, @Ri	1100011i	1	1	1	(A) ↔ РПД(Ri)
28	Обмін молодшої тетради акумулятора з молодшою тетрадою байта РПД	XCHD A, @Ri	1101011i	1	1	1	$(A_{0–3}) \leftrightarrow РПД(Ri)_{0–3}$

4.12.4 Арифметичні команди

В наборі команд МК–ра (таблиця 4.4) є такі арифметичні операції: додавання, додавання з врахуванням прапорця перенесення, віднімання з позикою, множення і ділення, інкремент, декремент та десяткова корекція.

В АЛП проводяться дії над цілими числами. У таких операціях над двома операндами, як: додавання ADD, додавання з перенесенням ADDC та віднімання з позикою SUBB, акумулятор зберігає перший операнд і приймає результат операції.

Другим операндом може бути регістр обраного банку робочих регістрів, комірка внутрішньої пам’яті даних із непрямою та прямою адресацією або байт безпосередніх даних. Вказані операції впливають на прапорці: переповнення, перенесення, допоміжного перенесення і прапорець парності в слові стану процесора PSW.

Використання розряду перенесення дозволяє підвищити точність при виконанні операцій додавання ADDC і віднімання SUBB.

Виконання операцій додавання і віднімання з урахуванням знака може бути здійснене за допомогою прапорця переповнення OV регістра PSW. Прапорець допоміжного перенесення AC забезпечує виконання арифметичних операцій у двійково–десятковому коді і використовується командою десяткової корекції DAA.

Операції інкременту і декременту на прапорці не впливають.

Таблиця 4.4 – Група команд арифметичних операцій

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
1	Додавання до акумулятора регістра (n = 0..7)	ADD A, Rn	00101rrr	1	1	1	(A) ← (A) + (Rn)
2	Додавання до акумулятора байта, що адресується прямо	ADD A, ad	00100101	3	2	1	(A) ← (A) + (ad)
3	Додавання до акумулятора байта з РПД (i = 0,1)	ADD A, @Ri	0010011i	1	1	1	(A) ← (A) + РПД(Ri)
4	Додавання до акумулятора константи	ADD A, #D8	00100100	2	2	1	(A) ← (A) + D8
5	Додавання до акумулятора регістра і перенесення	ADDC A, Rn	00111rrr	1	1	1	(A) ← (A) + (Rn) + (C)
6	Додавання до акумулятора байта, що адресується прямо, і перенесення	ADDC A, ad	00110101	3	2	1	(A) ← (A) + (ad) + (C)
7	Додавання до акумулятора байта з РПД і перенесення	ADDC A, @Ri	0011011i	1	1	1	(A) ← (A) + РПД(Ri) + (C)
8	Додавання до акумулятора константи і перенесення	ADDC A, #D8	00110100	2	2	1	(A) ← (A) + D8 + (C)
9	Десяткова корекція акумулятора (після додавання двох операндів, що представлені в упакованому BCD–коді)	DA A	11010100	1	1	1	Якщо (A_{3-0})>9 V (AC)=1, то (A_{7-0}) ← (A_{7-0}) + 6; потім якщо (A_{7-4})>9 V (C)=1, то (A_{7-4}) ← (A_{7-4}) + 6
10	Віднімання від акумулятора регістра і позики	SUBB A, Rn	10011rrr	1	1	1	(A) ← (A) – (C) – (Rn)
11	Віднімання від акумулятора байта, що адресується прямо, і позики	SUBB A, ad	10010101	3	2	1	(A) ← (A) – (C) – (ad)
12	Віднімання від акумулятора байта РПД і позики	SUBB A, @Ri	1001011i	1	1	1	(A) ← (A) – (C) – РПД(Ri)
13	Віднімання від акумулятора константи і позики	SUBB A, #D8	10010100	2	2	1	(A) ← (A) – (C) – D8
14	Інкремент акумулятора	INC A	00000100	1	1	1	(A) ← (A) + 1
15	Інкремент регістра	INC Rn	00001rrr	1	1	1	(Rn) ← (Rn) + 1
16	Інкремент байта, що адресується прямо	INC ad	00000101	3	2	1	(ad) ← (ad) + 1
17	Інкремент байта в РПД	INC @Ri	0000011i	1	1	1	РПД(Ri) ← РПД(Ri) + 1
18	Інкремент покажчика даних	INC DPTR	10100011	1	1	2	(DPTR) ← (DPTR) + 1
19	Декремент акумулятора	DEC A	00010100	1	1	1	(A) ← (A) – 1
20	Декремент регістра	DEC Rn	00011rrr	1	1	1	(Rn) ← (Rn) – 1
21	Декремент байта, що адресується прямо	DEC ad	00010101	3	2	1	(ad) ← (ad) – 1
22	Декремент байта в РПД	DEC @Ri	0001011i	1	1	1	РПД(Ri) ← РПД(Ri) – 1
23	Множення акумулятора на регістр B	MUL AB	10100100	1	1	4	Якщо (A)x(B) > 255, то (B)(A) ← (A)x(B); якщо (A)x(B) < 255, то (A) ← (A)x(B)
24	Ділення акумулятора на регістр B	DIV AB	10000100	1	1	4	(A) ← (A)/(B) – ціла частина (B) ← R ((A)/(B)) – остача від ділення

4.12.5 Логічні команди

АЛП МК–ра дає можливість виконувати логічні операції над байтовими і бітовими операндами. В таблицю 4.5 включено команди виконання логічних операцій над вмістом 8–розрядних даних. Операції над бітовими операндами описані в таблиці 4.6.

Система команд МК–ра дозволяє реалізувати такі логічні операції: "І", "АБО", "ВИКЛЮЧНЕ АБО" над операндом в регістрі–акумуляторі А і байтом–джерелом. Другим операндом (байтом–джерелом) при цьому можуть бути регістр у вибраному банку робочих регістрів, регістр внутрішнього ОЗП, який адресується за допомогою непрямої адресації, комірки внутрішнього ОЗП, які адресуються прямо, регістри спеціального призначення, а також безпосередні операнди. Вказані логічні операції можуть бути реалізовані або на будь–якому регістрі внутрішнього ОЗП, який адресується прямо, або на регістрі спеціального призначення, беручи за другий операнд вміст акумулятора А або безпосередній операнд.

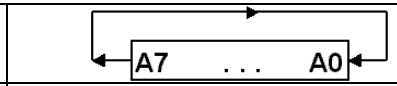
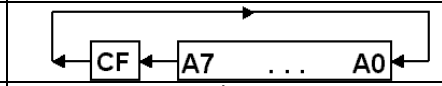
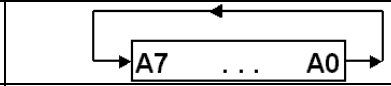
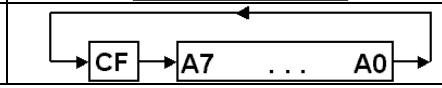
Існують логічні операції, що виконуються тільки в акумуляторі: скидання та інвертування усіх восьми розрядів А; циклічний зсув вліво та вправо; циклічний зсув вліво та вправо з урахуванням прапорця перенесення; обмін місцями старшої і молодшої тетрад акумулятора.

4.12.6 Операції з бітами

До складу МК–ра входить так званий "бітовий" процесор, який забезпечує виконання ряду операцій над бітами (таблиця 4.6).

Бітовий процесор є частиною архітектури МК–ра сімейства MCS–51 і його можна розглядати як незалежний процесор для побітової обробки. Бітовий процесор виконує свій набір команд, має свій ОЗП, що адресується побітово, і свій пристрій введення/виведення.

Таблиця 4.5 – Група команд логічних операцій

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
1	Логічне І акумулятора та регістра	ANL A, Rn	01011rrr	1	1	1	(A) ← (A) Λ (Rn)
2	Логічне І акумулятора та байта, що адресується прямо	ANL A, ad	01010101	3	2	1	(A) ← (A) Λ (ad)
3	Логічне І акумулятора та байта з РПД	ANL A, @Ri	0101011i	1	1	1	(A) ← (A) Λ РПД(Ri)
4	Логічне І акумулятора та константи	ANL A, #D8	01010100	2	2	1	(A) ← (A) Λ #D8
5	Логічне І байта, що адресується прямо, та акумулятора	ANL ad, A	01010010	3	2	1	(ad) ← (ad) Λ (A)
6	Логічне І байта, що адресується прямо, та константи	ANL ad, #D8	01010011	7	3	2	(ad) ← (ad) Λ #D8
7	Логічне АБО акумулятора та регістра	ORL A, Rn	01001rrr	1	1	1	(A) ← (A) V (Rn)
8	Логічне АБО акумулятора та байта, що адресується прямо	ORL A, ad	01000101	3	2	1	(A) ← (A) V (ad)
9	Логічне АБО акумулятора та байта з РПД	ORL A, @Ri	0100011i	1	1	1	(A) ← (A) V РПД(Ri)
10	Логічне АБО акумулятора та константи	ORL A, #D8	01000100	2	2	1	(A) ← (A) V D8
11	Логічне АБО байта, що адресується прямо, та акумулятора	ORL ad, A	01000010	3	2	1	(ad) ← (ad) V (A)
12	Логічне АБО байта, що адресується прямо, та константи	ORL ad, #D8	01000011	7	3	2	(ad) ← (ad) V #D8
13	Виключне АБО акумулятора та регістра	XRL A, Rn	01101rrr	1	1	1	(A) ← (A) ⊕ (Rn)
14	Виключне АБО акумулятора та байта, що адресується прямо	XRL A, ad	01100101	3	2	1	(A) ← (A) ⊕ (ad)
15	Виключне АБО акумулятора та байта РПД	XRL A, @Ri	0110011i	1	1	1	(A) ← (A) ⊕ РПД(Ri)
16	Виключне АБО акумулятора та константи	XRL A, #D8	01100100	2	2	1	(A) ← (A) ⊕ #D8
17	Виключне АБО байта, що адресується прямо, та акумулятора	XRL ad, A	01100010	3	2	1	(ad) ← (ad) ⊕ (A)
18	Виключне АБО байта, що адресується прямо, та константи	XRL ad, #D8	01100011	7	3	2	(ad) ← (ad) ⊕ #D8
19	Скидання акумулятора	CLR A	11100100	1	1	1	(A) ← 0
20	Інверсія акумулятора	CPL A	11110100	1	1	1	(A) ← ($\bar{A}$)
21	Зсув акумулятора вліво циклічний 	RL A	00100011	1	1	1	(A_{n+1}) ← (A_n), n = 0..6, (A_0) ← (A_7)
22	Зсув акумулятора вліво циклічний через перенесення 	RLC A	00110011	1	1	1	(A_{n+1})← (A_n), n=0..6, (A_0) ← (C), (C) ← (A_7)
23	Зсув акумулятора вправо циклічний 	RR A	00000011	1	1	1	(A_n) ← (A_{n+1}), n = 0..6, (A_7) ← (A_0)
24	Зсув акумулятора вправо циклічний через перенесення 	RRC A	00010011	1	1	1	(A_n) ← (A_{n+1}), n = 0..6, (A_7) ← (C), (C) ← (A_0)
25	Обмін місцями тетрад у акумуляторі	SWAP A	11000100	1	1	1	(A_{3-0}) ↔ (A_{7-4})

Таблиця 4.6 – Група команд операцій з бітами

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
1	Скидання перенесення	CLR C	11000011	1	1	1	(C) ← 0
2	Скидання біта, що адресується прямо	CLR bit	11000010	4	2	1	(b) ← 0
3	Встановлення перенесення	SETB C	11010011	1	1	1	(C) ← 1
4	Встановлення біта, що адресується прямо	SETB bit	11010010	4	2	1	(b) ← 1
5	Інверсія перенесення	CPL C	10110011	1	1	1	(C) ← $\overline{\mathbf{C}}$
6	Інверсія біта, що адресується прямо	CPL bit	10110010	4	2	1	(b) ← $\overline{\mathbf{b}}$
7	Логічне І перенесення та біта, що адресується прямо	ANL C, bit	10000010	4	2	2	(C) ← (C) Λ (b)
8	Логічне І перенесення та інверсії біта, що адресується прямо	ANL C, /bit	10110000	4	2	2	(C) ← (C) Λ ($\overline{\mathbf{b}}$)
9	Логічне АБО перенесення та біта, що адресується прямо	ORL C, bit	01110010	4	2	2	(C) ← (C) V (b)
10	Логічне АБО перенесення та інверсії біта, що адресується прямо	ORL C, /bit	10100000	4	2	2	(C) ← (C) V ($\overline{\mathbf{b}}$)
11	Пересилка біта, що адресується прямо, у перенесення	MOV C, bit	10100010	4	2	1	(C) ← (b)
12	Пересилка перенесення у біт, що адресується прямо	MOV bit, C	10010010	4	2	2	(b) ← (C)

Команди, що оперують із бітами, забезпечують пряму адресацію 128 бітів у шістнадцятьох комірках внутрішнього ОЗП (комірки з адресами 20H–2FH) і пряму побітову адресацію регістрів спеціального призначення, адреси яких кратні восьми: P0 (80H), TCON (88H), P1 (90H), SCON (98H), P2 (A0H), IE (A8H), P3 (B0H), IP (B8H), PSW (D0H), A (E0H), B (F0H).

Кожен з бітів, що адресуються прямо, може бути встановлений у "лог. 1", скинутий у "лог. 0", інвертований, або перевірений. Можуть бути реалізовані переходи: якщо біт встановлений; якщо біт не встановлений; перехід, якщо біт встановлений, із скиданням цього біта; біт може бути перезаписаний у (із) розряд(у) перенесення. Між будь–яким бітом, що адресується прямо, і прапорцем перенесення можуть бути виконані логічні операції "І", "АБО", а результат заноситься у розряд прапорця перенесення. Команди побітової обробки забезпечують реалізацію складних функцій комбінаторної логіки та оптимізацію програм користувача.

4.12.7 Команди передачі керування

До даної групи команд (таблиця 4.7) відносяться команди, що забезпечують умовні та безумовні переходи, виклик підпрограм і повернення з них, а також команда пустої операції NOP. У більшості команд використовується пряма адресація, тобто адреса переходу цілком (або його частина) залишається у самій команді передачі керування. Можна виділити три різновиди команд переходів за вказаною розрядністю адреси переходу.

Довгий перехід. Це перехід по всьому адресному простору ПП. В команді міститься повна 16–бітна адреса переходу ad16. Трьохбайтні команди довгого переходу містять у мнемоніці букву L (Long).

Таблиця 4.7 – Група команд передачі керування

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
1	Довгий перехід по повному обсязі пам'яті програм	LJMP ad16	00000010	12	3	2	(PC) ← ad16
2	Абсолютний перехід всередині сторінки у 2 Кбайт	AJMP ad11	a10a9a800001	6	2	2	(PC) ← (PC) + 2 (PC_{10-0}) ← ad11
3	Короткий відносний перехід всередині сторінки у 256 байт	SJMP rel	10000000	5	2	2	(PC) ← (PC) + 2 (PC) ← (PC) + rel
4	Непрямий перехід	JMP @A+DPTR	01110011	1	1	2	(PC) ← (A) + (DPTR)
5	Перехід, якщо акумулятор дорівнює нулю	JZ rel	01100000	5	2	2	(PC) ← (PC) + 2, якщо (A) = 0, тоді (PC) ← (PC) + rel
6	Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	01110000	5	2	2	(PC) ← (PC) + 2, якщо (A) ≠ 0, тоді (PC) ← (PC) + rel
7	Перехід, якщо перенесення дорівнює одиниці	JC rel	01000000	5	2	2	(PC) ← (PC) + 2, якщо (C) = 1, тоді (PC) ← (PC) + rel
8	Перехід, якщо перенесення дорівнює нулю	JNC rel	01010000	5	2	2	(PC) ← (PC) + 2, якщо (C) = 0, тоді (PC) ← (PC) + rel
9	Перехід, якщо біт дорівнює одиниці	JB bit, rel	00100000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 1, тоді (PC) ← (PC) + rel
10	Перехід, якщо біт дорівнює нулю	JNB bit, rel	00110000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 0, тоді (PC) ← (PC) + rel
11	Перехід, якщо біт встановлений, з наступним скиданням біта	JBC bit, rel	00010000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 1, тоді (b) ← 0 и (PC) ← (PC)
12	Декремент регістра і перехід, якщо не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC) ← (PC) + 2, (Rn) ← (Rn) –1, якщо (Rn) ≠ 0, то (PC)←(PC) +rel
13	Декремент байта, що адресується прямо, і перехід, якщо не нуль	DJNZ ad, rel	11010101	8	3	2	(PC) ← (PC) + 3, (ad) ← (ad) –1, якщо (ad) ≠ 0, то (PC) ←(PC) +rel
14	Порівняння акумулятора з байтом, що адресується прямо, і перехід, якщо не дорівнює	CJNE A, ad, rel	10110101	8	3	2	(PC) ← (PC) + 3, якщо (A) ≠ (ad), то (PC)←(PC)+rel, якщо(A)< (ad), то (C) ←1, інакше (C) ← 0

Продовження таблиці 4.7

№	Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
15	Порівняння акумулятора з константою і перехід, якщо не дорівнює	CJNE A, #D8, rel	10110100	10	3	2	(PC) ← (PC) + 3, якщо (A) ≠ #D8, то (PC) ← (PC) + rel, якщо(A) < D8, то (C) ← 1, інакше (C) ← 0
16	Порівняння регістра з константою і перехід, якщо не дорівнює	CJNE Rn, #D8, rel	10111rrr	10	3	2	(PC) ← (PC) + 3, якщо (Rn) ≠ #D8, то (PC) ← (PC) + rel,якщо(Rn) < D8, то (C) ← 1, інакше (C) ← 0
17	Порівняння байта в РПД з константою і перехід, якщо не дорівнює	CJNE @Ri, #D8, rel	1011011i	10	3	2	(PC) ← (PC) + 3, якщо РПД(Ri) ≠ #D8, то (PC) ← (PC) + rel, якщо РПД(Ri) < #D8, то (C) ←1, інакше (C) ← 0
18	Довгий виклик підпрограми	LCALL ad16	00010010	12	3	2	(PC) ← (PC) +3, (SP) ← (SP) + 1, РПД(SP) ← $(PC_{7–0})$, (SP) ← (SP) + 1, РПД(SP) ← $(PC_{8–15})$, (PC) ← ad16
19	Абсолютний виклик підпрограми в межах сторінки у 2 Кбайти	ACALL ad11	a10a9a810001	6	2	2	(PC) ← (PC) +2, (SP) ← (SP) + 1, РПД(SP) ← $(PC_{7–0})$, (SP)← (SP) + 1, РПД(SP) ← $(PC_{15–8})$, (PC) ← ad11
20	Повернення з підпрограми	RET	00100010	1	1	2	$(PC_{15–8})$ ←РПД(SP),(SP)←(SP)–1, $(PC_{7–0})$ ←РПД(SP),(SP) ← (SP)–1
21	Повернення з підпрограми обробки переривання	RETI	00110010	1	1	2	$(PC_{15–8})$ ←РПД(SP),(SP)←(SP)–1, $(PC_{7–0})$ ← РПД(SP),(SP) ←(SP)–1
22	Холоста команда	NOP	00000000	1	1	1	(PC) ← (PC) + 1

Примітка. Асемблер припускає використання узагальненого імені команд JMP і CALL, які в процесі трансляції замінюються оптимальними за форматом командами переходу (AJMP, SJMP, LJMP) або виклику (ACALL, LCALL).

Всього існує дві такі команди: LJMP – довгий перехід і LCALL – довгий виклик підпрограми. На практиці рідко виникає необхідність переходу в межах всього адресного простору і частіше використовуються скорочені команди переходу, що займають менше місця в пам'яті.

Абсолютний перехід. Це перехід в межах однієї сторінки пам'яті програм розміром 2048 байт. Такі команди містять тільки 11 молодших біт адреси переходу ad11. Команди абсолютного переходу мають формат 2 байти. Початкова буква мнемоніки – A (Absolute). При виконанні команди в обчисленій адресі наступної по порядку команди ((PC) = (PC) + 2) одинадцять молодших біт замінюються на ad 11 із тіла команди абсолютного переходу.

Відносний перехід. Цей короткий перехід дозволяє передати керування в межах від –128 до +127 байт відносно адреси наступної команди (команди, що йде наступною по порядку за командою відносного переходу).

Існує одна команда безумовного короткого відносного переходу SJMP (Short). Всі команди умовного переходу використовують даний метод адресації. Відносна адреса переходу rel міститься у другому байті команди.

Непрямий перехід. Команда JMP @A+DPTR дозволяє передавати керування за непрямою адресою. Ця команда зручна тим, що надає можливість організації переходу за адресою, яка обчислюється самою програмою і невідома при написанні вихідного тексту програми.

Умовні переходи. Розвинута система умовних переходів надає можливість здійснювати переходи за такими умовами: вміст акумулятора дорівнює нулю (JZ); вміст акумулятора не дорівнює нулю (JNZ); перенесення дорівнює одиниці (JC); перенесення дорівнює нулю (JNC);

біт, що адресується, дорівнює одиниці (JB); біт, що адресується, дорівнює нулю (JNB).

Для організації програмних циклів зручно користуватись командою DJNZ. Як лічильник циклів може використовуватись регістр або байт, що адресується прямо (наприклад, комірка РПД).

Команда порівняння CJNE ефективно використовується в процедурах очікування якоїсь події. Наприклад, команда

```
WAIT:   CJNE  A, P0, WAIT
```

буде виконуватись доти, поки на лініях порту 0 не встановиться інформація, що збігається із вмістом акумулятора.

Всі команди даної групи, за винятком CJNE, не впливають на прапорці. Команда CJNE встановлює прапорець C, якщо перший операнд виявляється меншим ніж другий.

Підпрограми. Для звернення до підпрограм необхідно використовувати команди виклику підпрограм (LCALL, ACALL). Ці команди, на відміну від команд переходу (LJMP, AJMP), зберігають у стеку адресу повернення в основну програму. Для повернення з підпрограми необхідно виконати команду RET. Команда RETI відрізняється від команди RET тим, що дозволяє переривання рівня, що обслуговується. Тому цю команду необхідно застосовувати наприкінці підпрограм, які викликані перериваннями.

4.12.8 Опис окремих команд

4.12.8.1 Команда ACALL <addr 11>

Команда "абсолютний виклик підпрограми" викликає безумовно підпрограму, яку розміщену за вказаною адресою (рисунок 4.7).

Асемблер:	ACALL <мітка>	
Код:	A10 A9 A8 1 0 0 0 1 A7 A6 A5 A4 A3 A2 A1 A0	
Час:	2 цикли	
Алгоритм:	(PC): = (PC) +2 (SP): = (SP) +1 ((SP)): = (PC [7–0]) (SP): = (SP) +1 ((SP)):=(PC ([15–8]) (PC [10–0]): = A10A9A8 II A7A6A5A4A3A2A1A0,	II–знак конкатенації (зчеплення)
Приклад:		;до виконання команди ACALL: ;(SP) = 07H ;мітка MT1 відповідає адресі: ;0345H, тобто ;(PC) = 0345H
	Команда ACALL MT1	;розташована за адресою 028DH, тобто ;(PC) = 028DH ;після виконання команди: ;(SP) = 09H, (pc) = 0345H, ;ОЗП [08] = 8FH, ОЗП [09] = 02H

Рисунок 4.7 – Приклад команди ACALL <мітка>

При цьому лічильник команд збільшується на 2 для отримання адреси наступної команди, після чого отримане 16–бітове значення PC поміщається в стек (спочатку йде молодший байт), і вміст покажчика стеку також збільшується на 2. Адреса переходу одержується за допомогою конкатенації п'яти старших біт лічильника команд PC (після збільшення його на два), 5–7 бітів коду операції та другого байта команди.

4.12.8.2 Команда ADD A, <байт джерело>

Ця команда підсумовує вміст акумулятора A з вмістом байта–джерела, залишаючи результат в акумуляторі. При появі перенесення з розрядів 7 і 3, встановлюються відповідно прапорці перенесення (C) і додаткового перенесення (AC), в протилежному випадку ці прапорці скидаються. При підсумовуванні цілих чисел без знака прапорець перенесення "C" вказує на переповнення розрядної сітки для додатних

чисел. При підсумовуванні цілих чисел зі знаком встановлення прапорця (OV) вказує на переповнення розрядної сітки, коли результат операції не вкладається у діапазон: – 128…+127. Прапорець переповнення (OV) встановлюється, якщо є перенесення з біта 6 і немає перенесення з біта 7, або є перенесення з біта 7 і немає з біта 6, у протилежному випадку прапорець (OV) скидається. Для команди додавання дозволені наступні режими адресації байта джерела:

1) регістрова;

2) непряма;

3) пряма;

4) безпосередня.

Приклади команди зображено на рисунку 4.8.

4.12.8.3 Команда ADDC A, <байт–джерело>

Ця команда ("підсумовування з перенесенням") (рисунок 4.9) підсумовує вміст байта–джерела, прапорець перенесення і вміст акумулятора A, залишаючи результат в акумуляторі. При цьому прапорці перенесення і додаткового перенесення встановлюються, якщо є перенесення з біта 7 або біта 3, і скидаються в протилежному випадку.

При підсумовуванні цілих чисел без знака на переповнення розрядної сітки вказує прапорець перенесення (C). Прапорець переповнення (OV) встановлюється якщо є перенесення з біта 6 і немає перенесення з біта 7, або є перенесення з біта 7 і немає з біта 6, у протилежному випадку (OV) скидається. При підсумовуванні цілих чисел зі знаком (OV) вказує на від’ємну величину, отриману при додаванні двох додатних операндів або на додатну суму від двох від’ємних операндів.

1) Асемблер:	ADD A, Rn, де	n = 0–7
Код:	0 0 1 0 1 r r r, де	r r r = 000–111
Час:	1 цикл	
Алгоритм:	(A):=(A)+(Rn), де C:=X, OV:=X, AC:=X	n=0–7 X=(0 або 1)
Приклад:	 ADD A, R6	;(A)=C3H, R6=AAH ;(A)=6DH, (R6)=AAH, (AC)=0 ;(C)=1, (OV)=1
2) Асемблер	ADD A, @Ri, де	i=0,1
Код:	0 0 1 0 0 1 1 i, де	i=0,1
Час:	1 цикл	
Алгоритм:	(A):=(A)+((Ri)), де C:=X, OV:=X, AC:=X,	i=0,1 X=0 або 1
Приклад:	 ADD A, @R1	;(A)=95H, (R1)=31H, ;(ОЗП[31])=4CH ;(A)=E1H, (ОЗП[31])=4CH, ;(C)=0, (OV)=0, (AC)=1
3) Асемблер:	ADD A, <direct>	
Код:	0 0 1 0 0 1 0 1	direct address
Час:	1 цикл	
Алгоритм:	(A):=(A)+(direct) C:=X,OV:=X, AC:=X	 X=0 або 1
Приклад:	 ADD A, 90H	;(A)=77H, ;(ОЗП[90])=FFH, ;(A)=76H, ;(ОЗП[90])=FFH, ;(C)=1, (OV)=0, (AC)=1
4) Асемблер:	ADD A,<#data>	
Код:	0 0 1 0 0 1 0 0	#data
Час:	1 цикл	
Алгоритм:	(A):=(A)+#data C:=X, OV:=X, AC:=X	 X=0 або 1
Приклад:	 ADD A, #0D3H	;(A)=09H, ;(A)=DCH, ;(C)=0,(OV)=0, (AC)=0

Рисунок 4.8 – Приклади команди ADD A, <байт джерело>

1) Асемблер:	ADDC A, Rn, де	n=0–7
Код:	0 0 1 1 1 r r r, де	r r r =000–111
Час:	1 цикл	
Алгоритм:	(A):=(A)+(Rn)+(C), де	n=0–7
	C:=X, OV:=X, AC:=X	X=(0 або 1)
Приклад:		;(A)=B2H,R3=99H,
		;(C)=1
	ADDC A, R3	;(A)=4CH, (R3)=99H, (AC)=0
		;(C)=1, (OV)=1
2) Асемблер	ADDC A, @Ri, де	i=0,1
Код:	0 0 1 1 0 1 1 i, де	i=0,1
Час:	1 цикл	
Алгоритм:	(A):=(A)+((Ri))+(C), де	i=0,1
	C:=X, OV:=X, AC:=X,	X=0 або 1
Приклад:		;(A)=D5H,(R0)=3AH,
		;(ОЗП[3A])=1AH,(C)=1
	ADDC A, @R0	;(A)=F0H, (ОЗП[3A])=1AH,
		;(C)=0, (OV)=0, (AC)=1
3) Асемблер:	ADDC A, <direct>	
Код:	0 0 1 1 0 1 0 1	direct address
Час:	1 цикл	
Алгоритм:	(A):=(A)+(direct)+(C)	
	C:=X, OV:=X, AC:=X	X=0 або 1
Приклад:		;(A)=11H, (ОЗП[80])=DFH,
		;(C)=1,
	ADDC A, 80H	;(A)=F1H,
		;(C)=1, (OV)=0, (AC)=1
4) Асемблер:	ADDC A,<#data>	
Код:	0 0 1 1 0 1 0 0	#data
Час:	1 цикл	
Алгоритм:	(A):=(A)+(direct)+(C)	
	C:=X, OV:=X, AC:=X	X=0 або 1
Приклад:		;(A)=55H, (C)=1
	ADDC A, #55H	;(A)=AAH,
		;(C)=0, (OV)=1, (AC)=0

Рисунок 4.9 – Приклади команди ADDC A, <байт–джерело>

Для команди ADDC A, <байт–джерело> дозволені наступні режими адресації байта–джерела:

1) регістрова;

2) непряма;

3) пряма;

4) безпосередня.

4.12.8.4 Команда AJMP <addr11>

Команда "абсолютний перехід" (рисунок 4.10) передає керування за вказаною адресою, яка отримується при конкатенації п'яти старших біт лічильника команд PC (після збільшення його на два), 5–7 бітів коду операції та другого байта команди. Адреса переходу повинна знаходиться всередині однієї сторінки об'ємом 2 Кбайт пам'яті програми, яка визначається п'ятьма старшими бітами лічильника команд.

```
Асемблер    AJMP <мітка>
Код         A10 A9 A8  0  0  0  0  1
            A7  A6  A5 A4 A3 A2 A1 A0
Час         2 цикли
Приклад:                              ;(PC)=028FH
                                      ;Мітці MT2 відповідає адреса 034AH
            AJMP MT2                  ;(PC)=034AH
```

Рисунок 4.10 – Приклад команди AJMP <мітка>

4.12.8.5 Команда ANL <байт призначення>, <байт джерело>

Команда "логічне "І" (рисунок 4.11) для змінних–байтів" виконує операцію логічного "І" над бітами вказаних змінних і поміщає результат в байт призначення. Ця операція не впливає на стан прапорців.

Два операнди забезпечують наступні комбінації шести режимів адресації:

– байтом призначення є акумулятор (A):

1) регістрова;

2) пряма;

3) непряма;

4) безпосередня;

– байтом призначення є пряма адреса (Direct):

5) неявна (акумулятор);

6) безпосередня (байт–джерело дорівнює константі).

Примітка. Якщо команда ANL застосовується для зміни вмісту порту, то значення, що використовується в якості даних порту, буде зчитуватися з "защіпки" порту, а не з виводів мікроконтролера.

1) Асемблер:	ANL A, Rn, де	де n=0–7
Код	01011 rrr, де	де rrr=000–111
Час	1 цикл	
Приклад		;(A)=FEH,(R2)=C5H
	ANL A, R2	;(A)=C4H,(R2)=C5H
2) Асемблер:	ANL A, <direct>	
Код	01010101 direct adres	
Час	1 цикл	
Приклад		;(A)=A3H,(PSW)=86H
	ANL A, PSW	;(A)=82H,(PSW)=86H
3) Асемблер	ANL A, @Ri, де	де i=0,1
Код	0101011i, де	де i=0,1
Час	1 Цикл	
Приклад		;(A)=BCH, (ОЗП[35])=47H,
		;(R0)=35H,
	ANL A, @R0	;(A)=04H, (ОЗП[35])=47H
4) Асемблер	ANL A, #data	
Код	01010100 #data8	
Час	1 цикл	
Приклад		;(A)=36H
	ANL A, #0DDH	;(A)=14H
5) Асемблер	ANL <direct>, A	
Код	0 1 0 1 0 0 1 0	direct address
Час	1 цикл	
Приклад		;(A)=55H,(P2)=AAh
	AN P2, A	;(P2)=00h, A=55h
6) Асемблер	ANL <direct>,#data	
Код	0 1 0 1 0 0 1 1	direct address #data8
Час	2 цикли	
Приклад		;(P1)=FFh
	ANL P1, #73h	;(P1)=73h

Рисунок 4.11 – Приклад команди ANL <байт призначення>, <байт джерело>

4.12.8.6 Команда ANL C, <біт джерело>

Команда "логічне "І" для змінних–бітів" (рисунок 4.12) виконує операцію логічного "І" над зазначеними бітами. Якщо біт–джерело дорівнює "0", то відбувається скидання прапорця перенесення, у протилежному випадку прапорець перенесення не змінює поточного значення. "/" перед операндом у мові асемблера вказує на те, що в якості значення використовується логічне заперечення адресованого біта, проте сам біт джерела при цьому не змінюється. На інші прапорці ця команда не впливає.

Для операнда–джерела дозволена тільки пряма адресація бітів.

1) Асемблер:	ANL C, <bit>	
Код:	10000010	bit address
Час:	2 цикли	
Приклад:		;(C)=1, P1[0]=0
	ANL C, P1.0	;(C)=0, P1[0]=0
2) Асемблер:	ANL C, </bit>	
Код:	10110000	bit address
Час:	2 цикли	
Приклад:		;(C)=1, (AC)=0
	ANL C, /AC	;(C)=1, (AC)=0

Риснок 4.12 – Приклад команди ANL C, <біт джерело>

4.12.8.7 Команда CJNE <байт призначення>, <байт джерело>, <зміщення>

Команда "порівняння і перехід, якщо не дорівнює" (рисунок 4.13) порівнює значення перших двох операндів і виконує розгалуження, якщо операнди не рівні.

Адреса переходу (розгалуження) обчислюється за допомогою підсумовування значення (зі знаком), зазначеного в останньому байті команди, з вмістом лічильника команд після збільшення його на три.

1) Асемблер:	CJNE A, <direct>, мітка>	
Код:	10110101	direct address rel8
Час:	2 цикли	
Приклад:		;(A)=97h, (P2)=F0h, (C)=0
	CJNE A, P2, MT3	;(A)=97h, (P2)=F0h, (C)=1
	.....	;Адреса, що відповідає мітці MT3
	MT3: CLR A	обчислюється, як
		;(PC):=(PC)+3+(rel8)
2) Асемблер:	CJNE A, #data, <мітка>	
Код:	10110100	#data8 rel8
Час:	2 цикли	
Приклад:		;(A)=FCh, C=1
	CJNE A, #0BFh, MT4	
	.....	;(A)=FDh, C=0
	MT4: INC A	;(PC):=(PC)+3+(rel8)
3) Асемблер:	CJNE Rn, #data, <мітка>,	де n=0–7
Код:	10111 rrr	#data8 rel8, де rrr=000–111
Час:	2 цикли	
Приклад:		;(R7)=80H, (C)=0
	CJNE A, #81h, MT5	
	.....	;(R7)=80H,(C)=1,
	MT5: NOP	;(PC):=(PC)+3+(rel8)
4) Асемблер:	CJNE @Ri, #data, <мітка>	де i=0,1
Код:	1011011i	#data8 rel8
Час:	2 цикли	
Приклад:		;(R0)=41h, (C)=1, (ОЗП[41])=57H
	CJNE @R0, #29, MT6	
	.....	
	MT6: DEC R0	;(ОЗП[41])=57H, (C)=0
		; (PC):=(PC)+3+(rel8)

Рисунок 4.13 – Приклад команди CJNE <байт призначення>, <байт джерело>, <зміщення>

Прапорець перенесення "C" встановлюється в "1", якщо значення цілого без знака <байта призначення> менше, ніж значення цілого без знака <байта джерела>, в протилежному випадку перенесення скидається (якщо значення операндів рівні, прапорець перенесення скидається). Ця команда не впливає на операнди.

Операнди, що стоять в команді, забезпечують комбінації чотирьох режимів адресації:

– якщо байтом призначення є акумулятор:

1) пряма,

2) безпосередня;

– якщо байтом призначення є РЗП або комірка ОЗП з непрямою адресацією:

3) безпосередня та регістрова,

4) безпосередня та непряма.

Приклад команди зображено на рисунку 4.13.

4.12.8.8 Команда CLR A

Команда "скидання акумулятора" (рисунок 4.14) скидає (обнуляє) вміст акумулятора А. На прапорці команда не впливає.

Асемблер:	CLR A	
Код:	11100100	
Час:	1 цикл	
Приклад:		;(A)=6Dh, (C)=0, (AC)=1
	CLR A	;(A)=00h, (C)=0, (AC)=1

Рисунок 4.14 – Приклад команди CLR A

4.12.8.9 Команда CLR <bit>

Команда "скидання біта" (рисунок 4.15) скидає вказаний біт в нуль. Ця команда працює з прапорцем перенесення "С" або будь–яким бітом з прямою адресацією.

1) Асемблер:	CLR C	
Код:	11000011	
Час:	1 цикл	
Приклад:		;(C)=1
	CLR C	;(C)=0
2) Асемблер:	CLR <bit>	
Код:	11000010	bit address
Час:	1 цикл	
Приклад:		;(P1)=5EH(01011110B)
	CLR P1.3	;(P1)=56H(01010110B)

Рисунок 4.15 – Приклад команди CLR <bit>

4.12.8.10 Команда CPL A

Команда "інверсія акумулятора" (рисунок 4.16) кожен біт акумулятора інвертує (змінює на протилежний). Біти, які містили "одиниці", після цієї команди будуть містити "нулі", і навпаки. На прапорці ця операція не впливає.

Асемблер:	CPL A	
Код:	11110100	
Час:	1 цикл	
Приклад:		;(A)=65H(01100101B)
	CPL A	;(A)=9AH(10011010B)

Рисунок 4.16 – Приклад команди CPL A

4.12.8.11 Команда CPL <bit>

Команда "інверсія біта" (рисунок 4.17) інвертує (змінює на протилежне значення) вказаний біт. Біт, який був "одиницею", змінюється в "нуль" і навпаки. Команда CPL може працювати з прапорцем перенесення або з будь–яким бітом, який адресується прямо. На інші прапорці команда не впливає.

Примітка. Якщо ця команда використовується для зміни інформації на виході порту, значення, яке використовується як вихідні дані, зчитується з "защіпки" порту, а не з виводів мікроконтролера.

1) Асемблер:	CPL <bit>	
Код:	10110010	bit address
Час:	1 цикл	
Приклад:		;(P1)=39H(00111001B)
	CPL P1.1	
	CPL P1.3	;(P1)=33H(00110011B)
2) Асемблер:	CPL C	
Код:	10110011	
Час:	1 цикл	
Приклад:		;(C)=0, (AC)=1, (OV)=0
	CPL C	;(C)=1,(AC)=1,(OV)=0

Рисунок 4.17 – Приклад команди CPL <bit>

4.12.8.12 Команда DA A

Команда "десяткова корекція акумулятора для додавання" (рисунок 4.18) впорядковує 8–бітову величину в акумуляторі після виконаної раніше команди додавання двох змінних (кожна в упакованому двійково–десятковому форматі).

Асемблер:	DA A	
Код:	11010100	
Час:	1 цикл	
Приклади:	а) ADDC A, R3	;(A)=56H,(R3)=67H,(C)=1
	DA A	;(A)=24H, (R3)=67H, (C)=1
	б) ADD A, #99H	;(A)=30H, (C)=0
	DA A	;(A)=29, (C)=1

Рисунок 4.18 – Приклад команди DA A

Для виконання додавання може використовуватись будь–яка з типів команд ADD або ADDC. Якщо значення бітів 3–0 акумулятора (A) перевищує 9 (XXXX 1010–XXXX 1111) або, якщо прапорець AC дорівнює

"1", то до вмісту (А) додається 06, при цьому отримується відповідна двійково–десяткова цифра в молодшому напівбайті. Внутрішнє побітове додавання встановлює прапорець перенесення, якщо перенесення з поля молодших чотирьох біт поширюється через всі старші біти, а в протилежному випадку – не змінює прапорець перенесення. Якщо після цього прапорець перенесення дорівнює "1", або якщо значення чотирьох старших біт (7–4) перевищує 9 (1010 XXXX – 1111 XXXX), значення цих старших біт збільшується на 6, створюючи відповідну двійково–десяткову цифру у старшому напівбайті. І знову при цьому прапорець перенесення встановлюється, якщо перенесення виходить із старших бітів, але не змінюється в протилежному випадку. Таким чином, прапорець перенесення вказує на те, що сума двох вихідних двійково–десяткових змінних більша, ніж 99. Ця команда виконує десяткове перетворення за допомогою додавання 06, 60, 66 з вмістом акумулятора в залежності від початкового стану акумулятора і слова стану програми (PSW).

Примітка. Команда DA A не може просто перетворити шістнадцяткове значення в акумуляторі в двійково–десяткове представлення, і не застосовується, наприклад, для десяткового віднімання.

4.12.8.13 Команда DEC <байт>

Команда "декремент" (рисунок 4.19) виконує віднімання "1" із зазначеного операнда. Початкове значення 00H перейде в FFH. Команда DEC не впливає на прапорці. Цією командою допускається чотири режими адресації операнда:

1) неявна (акумулятор);

2) регістрова;

3) пряма;

4) непряма.

1) Асемблер	DEC A	
Код	00010100	
Час	1 цикл	
Приклад		;(A)=11H, (C)=1, (AC)=1
	DEC A	;(A)=10H, (C)=1,(AC)=1
2) Асемблер	DEC Rn,	де n = 0–7
Код	00011 r r r,	де r r r = 000–111
Час	1 цикл	
Приклад		;(R1)=7FH
		;(ОЗП[7F])=40H, (ОЗП[7F])=00H
	DEC @R1	
	DEC R1	
	DEC @R1	;(R1)=7EH,
		;(ОЗП[7F])=3FH, (ОЗП[7FH])=FFH
3) Асемблер	DEC <direct>	
Код	00010101	direct address
Час	1 цикл	
Приклад		;(SCON)=A0H, (C)=1, (AC)=1
	DEC SCON	;(SCON)=9FH, (C)=1, (AC)=1
4) Асемблер	DEC @Ri,	де i=0, 1
Код	0001011i	
Час	1 цикл	
Приклад		;(R1)=7FH,
		;(ОЗП[7F])=40H, (ОЗП[7F])=00H
	DEC @R1	
	DEC R1	
	DEC @R1	;(R1)=7EH,
		;(ОЗП[7F])=3FH, (ОЗП[7F])=FFH

Рисунок 4.19 – Приклад команди DEC <байт>

Примітка. Якщо ця команда використовується для зміни інформації на виході порту, значення, яке використовується як вихідні дані, зчитується з "защіпки" порту, а не з виводів мікроконтролера.

4.12.8.14 Команда DIV A B

Команда "ділення" (рисунок 4.20) ділить 8–бітове ціле без знаку з акумулятора A на 8–бітове ціле без знаку в регістрі B. Акумулятору присвоюється ціла частина частки (старші розряди), а регістру B –

залишок. Прапорці перенесення (C) та переповнення (OV) скидаються. Якщо (A)<(B), то прапорець додаткового перенесення (AC) не скидається. Прапорець перенесення скидається в будь–якому випадку.

Асемблер	DIV AB
Код	10000100
Час	4 цикли
Приклад	Нехай акумулятор містить число 251 (0FBH або 11111011B), а регістр B – число 18 (12H або 00010010B), Після виконання команди DIV AB в акумуляторі буде число 13 (0DH або 00001101B), а в регістрі B – число 17 (11H або 00010001B), оскільки 251=(13*18)+17. Прапорці C і OV будуть скинуті.

Рисунок 4.20 – Приклад команди DIV A B

Примітка. Якщо B містить 00, то після команди DIV вміст акумулятора A і регістра B будуть не визначені. Прапорець перенесення скидається, а прапорець переповнення встановлюється в "1".

4.12.8.15 Команда DJNZ <байт>, <зміщення>

Команда "декремент і перехід, якщо дорівнює нулю" (рисунок 4.21) виконує віднімання "1" з вказаної комірки і здійснює розгалуження за обчислюваною адресою, якщо результат не дорівнює нулю. Початкове значення 00H перейде в FFH. Адреса переходу (розгалуження) обчислюється додаванням значення зміщення (зі знаком), зазначеного в останньому байті команди, з вмістом лічильника команд, збільшеним на довжину команди DJNZ. На прапорці ця команда не впливає і допускає такі режими адресації:

1) регістрова;

2) пряма.

1) Асемблер:	DJNZ Rn,<мітка>,	де n=0–7
Код:	11011rrr	re18, де rrr = 000–111
Час:	2 цикли	
Приклад:		;(R2)=08h, (P1)=FFH (11111111B)
	LAB4: CPL P1.7 DJNZ R2, LAB4	;(R2)={07–00} ;ця послідовність команд ;переключає P1.7 вісім разів ;і призводить до виникнення ;чотирьох імпульсів на виводі МК, ;який відповідає біту P1.7
2) Асемблер:	DJNZ <direct>, <мітка>	
Код:	11010101	direct addres re18
Час:	2 цикли	
Приклад:		;(ОЗП[40])=01H, (ОЗП[50])=80H, ;(ОЗП[60])=25H
	DJNZ 40H, LAB1	;(ОЗП[40]):=00H
	DJNZ 50H, LAB2	;(ОЗП[50]):=7FH
	DJNZ 60H, LAB3	;(ОЗП[60]):=25H
	...	
	LAB1: CLR A	
	...	
	LAB2: DEC R1	;здійснюється перехід на мітку LAB2

Рисунок 4.21 – Приклад команди DJNZ <байт>, <зміщення>

Примітка. Якщо команда DJNZ використовується для зміни виходу порту, значення, яке використовується як операнд, зчитується з "защіпки" порту, а не з виводів МК.

4.12.8.16 Команда INC <байт>

Команда "інкремент" (рисунок 4.22) виконує додавання "1" до зазначеної змінної і не впливає на прапорці. Початкове значення FFH перейде в 00H. Ця команда допускає 4 режими адресації:

1) неявна (акумулятор);

2) регістрова;

3) пряма;

4) непряма.

1) Асемблер :	INC A	
Код:	00000100	
Час:	1 цикл	
Приклад:		;(A)=1FH, (AC)=0
	INC A	;(A)=20H, (AC)=0
2) Асемблер :	INC Rn,	де n=0–7
Код:	00001rrr,	де rrr=000–111
Час:	1 цикл	
Приклад:		;(R4)=FFH, (C)=0, (AC)=0
	INC R4	;(R4)=00H, (C)=0, (AC)=0
3) Асемблер :	INC <direct>	
Код:	00000101	[direct address]
Час:	1 цикл	
Приклад:		;(ОЗП[43])=22H
	INC 43H	;(ОЗП[43])=23H
4) Асемблер :	INC @Ri,	де i=0,1
Код:	0000011i,	де i=0,1
Час:	1 цикл	
Приклад:		;(R1)=41H, (ОЗП[41])=4FH, (AC)=0
	INC @R1	;(R1)=41H, (ОЗП[41])=50H, (AC)=0

Рисунок 4.22 – Приклад команди INC <байт>

Примітка. При використанні команди INC для зміни вмісту порту, величина, використовувана як операнд, зчитується з "защіпки" порту, а не з виводів МК.

4.12.8.17 Команда INC DPTR

Команда "інкремент покажчика даних" (рисунок 4.23) виконує інкремент (додавання "1") вмісту 16–бітового покажчика даних (DPTR). Додавання "1" здійснюється до 16 бітів, причому переповнення молодшого байта покажчика даних (DPL) з FFH в 00H призводить до інкременту старшого байта покажчика даних (DPH). На прапорці ця команда не впливає.

Асемблер :	INC DPTR	
Код:	10100011	
Час:	2 цикли	
Приклад:		;(DPH)=12H, (DPL)=FEH
	INC DPTR	
	INC DPTR	
	INC DPTR	;(DPH)=13H, (DPL)=01H

Рисунок 4.23 – Приклад команди INC DPTR

4.12.8.18 Команда JB <bit>, <rel8>

Команда "перехід, якщо біт встановлено" (рисунок 4.24) виконує перехід за адресою розгалуження, якщо зазначений біт дорівнює "1", в протилежному випадку виконується наступна команда. Адреса розгалуження обчислюється за допомогою додавання відносного зміщення зі знаком у третьому байті команди (rel8) до вмісту лічильника команд після додавання до нього 3. Біт, який перевіряється, не змінюється. Ця команда на прапорці не впливає.

Асемблер :	JB (bit), <мітка>	
Код:	00100000	bit address rel8
Час:	2 цикли	
Приклад:		;(A)=96H (10010110B)
	JB ACC.2, LAB5	
	. . .	
	LAB5: INC A	

Рисунок 4.24 – Приклад команди JB <bit>, <rel8>

4.12.8.19 Команда JBC <bit>, <rel8>

Команда "перехід, якщо біт встановлено і скидання цього біта" (рисунок 4.25), виконує розгалуження за обчислюваною адресою, якщо біт дорівнює "1". В протилежному випадку виконується наступна за JBC команда. У будь–якому випадку вказаний біт скидається. Адреса переходу обчислюється додаванням відносного зміщення зі знаком у третьому байті

команди (rel8) і вмісту лічильника команд після додавання до нього 3. Ця команда не впливає на прапорці.

```
Асемблер:   JBC <bit>,<мітка>
Код:        00010000            bit address    rel8
Час:        2 цикли
Приклад:                        ;(A)=76h (0111 0110b)
            JBC ACC.3,LAB6      ;Переходу на LAB6 немає, тому що (A[3])=0
            JBC ACC.2,LAB7      ;(A)=72h (01110010b) і перехід на адресу,
                                ;яка відповідає мітці LAB7
```

Рисунок 4.25 – Приклад команди JBC <bit>, <rel8>

Примітка. Якщо ця команда використовується для перевірки біт порту, то значення, що використовується як операнд, зчитується з "защіпки" порту, а не з виводів МК.

4.12.8.20 Команда JC <rel8>

Команда "перехід, якщо перенесення встановлено" (рисунок 4.26) виконує розгалуження за адресою, якщо прапорець перенесення дорівнює "1", в протилежному випадку виконується наступна команда. Адреса розгалуження обчислюється за допомогою додавання відносного зміщення зі знаком у другому байті команди (rel8) і вмісту лічильника команд, після додавання до нього 2. Ця команда не впливає на прапорці.

```
Асемблер:   JC  <мітка>
Код:        01000000            rel8
Час:        2 цикли
Приклад:                        ;(C)=0
                JC LAB8         ;немає переходу на мітку LAB8
                CPL C           ;(C):=1
            LAB8: JC LAB9       ;перехід на мітку LAB9, бо (C)=1
                  .....
            LAB9: NOP
```

Рисунок 4.26 – Приклад команди JC <rel8>

4.12.8.21 Команда JMP @A + DPTR

Команда "непрямий перехід" (рисунок 4.27) підсумовує восьмибітовий вміст акумулятора без знаку з 16–бітовим покажчиком даних (DPTR) і завантажує отриманий результат у лічильник команд, вміст якого є адресою для вибірки наступної команди. 16–бітове додавання виконується за модулем 2^{16}, перенесення з молодших восьми біт поширюється на старші біти програмного лічильника. Вміст акумулятора і покажчика даних не змінюється.

Асемблер:	JMP @A+DPTR	
Код:	01110011	
Час:	2 цикли	
Приклад:		;(PC)=034Eh,
		;(A)=86h,
		;(DPTR)=0329h
	JMP @A+DPTR	;(PC)=03AFh,
		;(A)=86h,
		;(DPTR)=0329h

Рисунок 4.27 – Приклад команди JMP @A + DPTR

4.12.8.22 Команда JNB <bit>, <rel8>

Команда "перехід, якщо біт не встановлено" (рисунок 4.28) виконує розгалуження за адресою, якщо зазначений біт дорівнює "0", в протилежному випадку виконується наступна команда. Адреса розгалуження обчислюється за допомогою додавання відносного зміщення зі знаком у третьому байті команди (rel8) і вмісту лічильника команд, після додавання до нього 3. Біт, який перевіряється, не змінюється. Ця команда не впливає на прапорці.

```
Асемблер:   JNB <bit>,<мітка>
Код:        00110000                    bit address rel8
Час:        2 цикли
Приклад:                                ;(P2)=CAH
                                        ;(11001010b),
                                        ;(A)=56h (0101 0110b)
                JNB P1.3, LAB10         ;немає переходу на LAB10
                JNB ACC.3, LAB11        ;перехід на мітку LAB11
                  .......
            LAB11: INC A
```

Рисунок 4.28 – Приклад команди JNB <bit>, <rel8>

4.12.8.23 Команда JNC <rel8>

Команда "перехід, якщо перенесення не встановлено" (рисунок 4.29) виконує розгалуження за адресою, якщо прапорець перенесення дорівнює "0", в протилежному випадку виконується наступна команда. Адреса розгалуження обчислюється за допомогою додавання відносного зміщення зі знаком у другому байті команди (rel8) і вмісту лічильника команд, після додавання до нього 2. Біт, який перевіряється, не змінюється. Ця команда не впливає на прапорці.

```
Асемблер:   JNC <мітка>
Код:        01010000                    rel8
Час:        2 цикли
Приклад:                                ;C=1
                JNC  LAB12              ;немає переходу на LAB12
                CPL  C
            LAB12:  JNC  LAB13          ;перехід на  мітку LAB13
```

Рисунок 4.29 – Приклад команди JNC <rel8>

4.12.8.24 Команда JNZ <rel8>

Команда "перехід, якщо вміст акумулятора не дорівнює нулю" (рисунок 4.30) виконує розгалуження за адресою, якщо хоча б один біт акумулятора дорівнює "1", в протилежному випадку виконується наступна

команда. Адреса розгалуження обчислюється додаванням відносного зміщення зі знаком у другому байті команди (rel8) і вмісту лічильника команд (PC), після додавання до нього 2. Вміст акумулятора не змінюється. Ця команда на прапорці не впливає.

```
Асемблер:   JNZ <мітка>
Код:        01110000                    rel8
Час:        2 цикли
Приклад:          JNC LAB14             ;немає переходу на LAB14
                  INC A
            LAB14: JNZ LAB15            ;перехід  на мітку LAB15
            LAB15: NOP
```

Рисунок 4.30 – Приклад команди JNZ <rel8>

4.12.8.25 Команда JZ <rel8>

Команда "перехід, якщо вміст акумулятора дорівнює "0"" (рисунок 4.31) виконує розгалуження за адресою, якщо всі біти акумулятора рівні "0", в протилежному випадку виконується наступна команда. Адреса розгалуження обчислюється додаванням відносного зміщення зі знаком у другому байті команди (rel8) і вмісту лічильника команд, після додавання до нього 2. Вміст акумулятора не змінюється. Ця команда на прапорці не впливає.

```
Асемблер:   JZ <мітка>
Код:        01100000                    rel8
Час:        2 цикли
Приклад:          JZ LAB16              ;немає переходу на LAB16
                  DEC A
            LAB16: JZ LAB17             ;перехід  на мітку LAB17
                   ......
            LAB17: CLR A
```

Рисунок 4.31 – Приклад команди JZ <rel8>

4.12.8.26 Команда LCALL <addr16>

Команда "довгий виклик" (рисунок 4.32) викликає підпрограму, що знаходиться за вказаною адресою.

Асемблер:	LCALL <мітка>	
Код:	00010010	addr[15–8] addr[7–0]
Час:	2 цикли	
Приклад:		;мітці PRN відповідає адреса 1234H
		;за адресою 0126H
		;знаходиться команда LCALL
	LCALL PRN	;(SP)=09H, (PC)=1234H,
		;(ОЗП[08])=26H, (ОЗП[09])=01H

Рисунок 4.32 – Приклад команди LCALL <addr16>

За командою LCALL до лічильника команд (PC) додається 3 для отримання адреси наступної команди і після цього отриманий 16–бітовий результат поміщається в СТЕК (спочатку йде молодший байт, а потім – старший), а вміст покажчика стеку (SP) збільшується на 2. Потім старший і молодший байти лічильника команд завантажуються відповідно другим і третім байтами команди LCALL. Виконання програми продовжується командою, яка знаходиться за отриманою адресою. Підпрограма, таким чином, може починатися в будь–якому місці адресного простору пам'яті програм обсягом до 64 Кбайт. Ця команда на прапорці не впливає.

4.12.8.27 Команда LJMP <addr16>

Команда "довгий перехід" (рисунок 4.33) виконує безумовний перехід за вказаною адресою, завантажуючи старший і молодший байти лічильника команд (PC) відповідно другим і третім байтами, що знаходяться в коді команди. Адреса переходу, таким чином, може перебувати за будь–якою адресою простору пам'яті програм в 64 Кбайт. Ця команда на прапорці не впливає.

Асемблер:	LJMP <мітка>	
Код:	00000010	addr[15–8] addr[7–0]
Час:	2 цикли	

Рисунок 4.33 – Приклад команди LJMP <addr16>

4.12.8.28 Команда MOV <байт призначення>, <байт джерела>

Команда "переслати змінну–байт" (рисунок 4.34) пересилає змінну–байт, зазначену в другому операнді в іншу комірку, яку вказано в першому операнді. Ця команда на прапорці та інші регістри не впливає. Вміст байта–джерела не змінюється. Ця команда допускає 15 комбінацій адресації байта–джерела і байта призначення.

1) Асемблер	MOV A, Rn, де	n=0–7
Код:	11101rrr, де	rrr=000–111
Час:	1 цикл	
Алгоритм:	(A):=(Rn)	
Приклад:		;(A)=FAh, (R4)=93h
	MOV A, R4	;(A)=FAh, (R4)=93h
2) Асемблер	MOV A,<direct>	
Код:	11100101	direct address
Час:	1 цикл	
Алгоритм:	(A):=(direct)	
Приклад:		;(A)=93h, (RAM[40])=10h, R0=40h
	MOV A, R4	;(A)=93h, (RAM[40])=10h, R0=40h
3) Асемблер	MOV A, @Ri, де	i=0,1
Код:	1110010i	
Час:	1 цикл	
Алгоритм:	(A):=((Ri))	
Приклад:		;(A)=10h, (RAM[41])=CAh, R0=41h
	MOV A, @R0	;(A)=CAh, (RAM[41])=CAh, R0=41h
4) Асемблер	MOV A, #data	
Код:	11100100	#data8
Час:	1 цикл	
Алгоритм:	(A):=<#data8>	
Приклад:		;(A)=c9h
	MOV A, #37h	;(A)=37h

Рисунок 4.34 – Приклад команди MOV <байт призначення>, <байт джерела>

5) Асемблер	MOV Rn, A, де	n=0–7
Код:	11111rrr, де	rrr=0–7
Час:	1 цикл	
Алгоритм:	(Rn):=(A)	
Приклад:		;(A)=38h, (R0)=42h
	MOV R0, A	;(A)=38h, (R0)=42h
6) Асемблер	MOV Rn,<direct>, де	n=0–7
Код:	10101rrr, де	rrr=0–7
Час:	2 цикл	
Алгоритм:	(Rn):=<direct>	
Приклад:		;(R0)=38h, (P2)=0F2h
	MOV R0, P2	;(R0)=0F2h
7) Асемблер	MOV Rn, #data, де	n=0–7
Код:	01111rrr, де	rrr=0–7
Час:	1 цикл	
Алгоритм:	(Rn):=#data	
Приклад:		;(R0)=38h
	MOV R0, #AAh	;(R0)=0AAh
8) Асемблер	MOV <direct>, A	
Код:	11110101	direct address
Час:	1 цикл	
Алгоритм:	(direct):=(A)	
Приклад:		;(P0)=38h, (A) = 10h
	MOV P0, A	;(P0)=10h
9) Асемблер	MOV <direct>, Rn, де	n=0–7
Код:	10001rrr, де	rrr=0–7
Час:	2 цикл	
Алгоритм:	(direct):=(Rn)	
Приклад:		;(P0)=38h, (A) = 10h
	MOV P0, A	;(P0)=10h
10) Асемблер	MOV <direct>,<direct>	
Код:	10000101	direct address, direct address
Час:	2 цикл	
Алгоритм:	(direct):=(direct)	
Приклад:		;(RAM[45])=33h, (RAM[48]) = 10h
	MOV 48h, 45h	;(RAM[48])=33h

Продовження рисунка 4.34

11) Асемблер	MOV <direct>,@Ri, де	i=0,1
Код:	1000011i	direct address
Час:	2 цикл	
Алгоритм:	(direct):=((Ri))	
Приклад:		;(R1)=49h, (RAM[49]) = 10h
	MOV 51h, @R1	;(RAM[51])=49h
12) Асемблер	MOV <direct>, #data	
Код:	10000101	direct address, direct address
Час:	2 цикл	
Алгоритм:	(direct):=(direct)	
Приклад:		;(RAM[45])=33h; (RAM[48]) = 10h
	MOV 48, 45	;(RAM[48])=33h
13) Асемблер	MOV <direct>, #data	
Код:	01110101	direct address, data
Час:	2 цикли	
Алгоритм:	(direct):=#data	
Приклад:		;(RAM[45h])=33h
	MOV 45h, #10h	;(RAM[45h])=10h
14) Асемблер	MOV @Ri,<direct>, де	i=0,1
Код:	1010011i	direct address
Час:	1 цикли	
Алгоритм:	((Ri)):=(direct)	
Приклад:		;(R0)=51h, (RAM[51h])=10, (P0)=5
	MOV @R0,<direct>	;(RAM[51h])=5h
15) Асемблер	MOV @Ri, #data, де	i=0,1
Код:	0111011i	data
Час:	1 цикли	
Алгоритм:	((Ri)):=#data	
Приклад:		;(R0)=51h, (RAM[51h])=10
	MOV @R0,#37	;(RAM[51h])=37

Продовження рисунка 4.34

4.12.8.29 Команда MOV <біт призначення>, <біт джерела>

Команда "переслати біт даних" (рисунок 4.35) бітову змінну, зазначену в другому байті, копіює в розряд, який вказано в першому операнді. Одним з операндів має бути прапорець перенесення C, а іншим може бути будь–який біт, до якого можлива пряма адресація.

1) Асемблер:	MOV C, <bit>	
Код:	10100010	bit address
Час:	1 цикл	
Алгоритм:	(C):=(bit)	
Приклад:		;(C)=0, (P3)=D5h (11010101b)
	MOV C, P3.0	;C:=1
	MOV C, P3.3	;C:=0
	MOV C, P3.7	;C:=1
2) Асемблер:	MOV <bit>, C	
Код:	10010010	bit address
Час:	2 цикли	
Алгоритм:		
Приклад:		;(C)=1, (P0)=20h (00100000b)
	MOV P0.1, C	
	MOV P0.2, C	
	MOV P0.3, C	;(C)=1, (P0)=2Eh (00101110b)

Рисунок 4.35 – Приклад команди MOV <біт призначення>, <біт джерела>

4.12.8.30 Команда MOV DPTR, # data16

Команда "завантажити покажчик даних 16–бітовою константою" (рисунок 4.36) завантажує покажчик даних DPTR 16–бітовою константою, зазначеною у другому і третьому байтах команди.

Асемблер:	MOV DPTR, #<data16>	
Код:	10010000	#data [15–8] #data [7–0]
Час:	2 цикли	
Алгоритм:	(DPTR):= #data [15–8], причому	
	DPH:= #data [15–8],	
	DPL:= #data [7–0]	
Приклад:		;(DPTR)=01FDh
	MOV DPTR, #1234h	;(DPTR)=1234h
		;(DPH)=12h
		;(DPL)=34h

Рисунок 4.36 – Приклад команди MOV DPTR, # data16

Другий байт команди завантажується в старший байт покажчика даних (DPH), а третій байт – в молодший байт покажчика даних (DPL). Ця команда на прапорці не впливає і є єдиною командою, яка одночасно завантажує 16 біт даних.

4.12.8.31 Команда MOVC A, @A + (<R16>)

<R16> – 16–розрядний регістр.

Команда "переслати байт з пам'яті програм" (рисунок 4.37) завантажує акумулятор байтом коду, або константою з пам'яті програми. Адреса зчитуваного байта обчислюється як сума 8–бітового вихідного вмісту акумулятора без знака та вмісту 16–бітового регістра. У якості 16–бітового регістра може бути:

1) покажчик даних DPTR,

2) лічильник команд PC.

1) Асемблер:	MOVC A,@A+DPTR	
Код:	10010011	
Час:	2 цикли	
Алгоритм:	(A):=((A)+ (DPTR))	
Приклад:		;(A)=1BH, (DPTR)=1020H
		;(ПЗП[103B])=48H
	MOVC A,@A+DPTR	;(A)=48H, (DPTR)=1020H
2) Асемблер:	MOVC A,@A+PC	
Код:	10000011	
Час:	2 цикли	
Алгоритм:	(A):=((A)+ (PC))	
Приклад:		;(A)=FAH, (PC)=0289H
		;(ПЗП[0384])=9BH
	MOVC A,@A+DPTR	;(A)=9BH, (PC)=028AH

Рисунок 4.37 – Приклад команди MOVC A, @A + (<R16>)

У випадку, коли використовується PC, він збільшується до адреси наступної команди перед тим, як його вміст підсумовується з вмістом акумулятора. 16–бітове додавання виконується так, що перенесення з молодших восьми біт може поширюватися через старші біти. Ця команда на прапорці не впливає.

4.12.8.32 Команда MOVX <байт призначення>, <байт джерела>

Команда "переслати в зовнішню пам'ять (із зовнішньої пам'яті) даних" (рисунок 4.38) пересилає дані між акумулятором та байтом

зовнішньої пам'яті даних. Є два типи команд, які відрізняються тим, що забезпечують 8–бітову або 16–бітову непряму адресу до зовнішнього ОЗП даних.

1) Асемблер:	MOVX A, @Ri,	де i=0,1
Код:	1110001i	
Час:	2 цикли	
Алгоритм:	(A):=((Ri))	
Приклад:		;(A)=32H, (R0)=83H,
		;комірка зовнішнього ОЗП
		;за адресою 83H містить B6H
		;(ПЗП[103B])=48H
	MOVX A, @R0	;(A)=B6H, (R0)=83H
2) Асемблер:	MOVX A, @DPTR	
Код:	11100000	
Час:	2 цикли	
Алгоритм:	(A):=((DPTR))	
Приклад:		;(A)=5CH
		;(DPTR)=1ABEH
		;комірка ОЗП за адресою 1ABEH містить 72H

Рисунок 4.38 – Приклад команди MOVX <байт призначення>, <байт джерела>

У першому випадку вміст R0 або R1 в поточному банку регістрів забезпечує 8–бітову адресу, яка мультиплексується з даними порту P0. Для адресації невеликого масиву зовнішнього ОЗП досить восьми біт адресації. Якщо застосовуються ОЗП, трохи більше ніж 256 байт, то для фіксації старших бітів адреси можна використовувати будь–які інші виходи портів, які переключаються командою, яка стоїть перед командою MOVX.

У другому випадку, при виконанні команди MOVX покажчик даних DPTR генерує 16–бітову адресу. Порт P2 виводить старші 8 біт адреси (DPH), а порт P0 мультиплексує молодші 8 біт адреси (DPL) з даними.

Ця форма є ефективною при доступі до великих масивів даних (до 64К), оскільки для встановлення портів виводення не потрібно додаткових команд.

4.12.8.33 Команда ORL <байт призначення>, <байт джерела>

Команда "логічне "АБО" для змінних–байтів" (рисунок 4.39) виконує операцію логічного "АБО" над бітами вказаних змінних, записуючи результат в байт призначення. Ця команда на прапорці не впливає.

1) Асемблер:	ORL A, Rn,	де n=0–7
Код:	01001 rrr,	де rrr=000–111
Час:	1 цикл	
Приклад:		;(A)=15H, (R5)=6CH
	ORL A, R5	;(A)=7DH, (R5)=6CH
2) Асемблер:	ORL A, <direct>	
Код:	01000101	direct adress
Час:	1 цикл	
Приклад:		;(A)=84H, (PSW)=C2H
	ORL A, PSW	;(A)=C6H, (PSW)=C2H
3) Асемблер:	ORL A, @Ri,	де i=0,1
Код:	0100011i	
Час:	1 цикл	
Приклад:		;(A)=52H, (R0)=6DH, (ОЗП[6D])=49H
	ORL A, @R0	;(A)=5BH, (ОЗП[6D])=49H
4) Асемблер:	ORL A, #<data>	
Код:	01000100	#data8
Час:	1 цикл	
Приклад:		;(A)=F0H
	ORL A, #0AH	;(A)=FAH
5) Асемблер:	ORL (direct), A	
Код:	01000010	direct adress
Час:	1 цикл	
Приклад:		;(A)=34H, (IP)=23H
	ORL IP, A	;(IP)=37H, (A)=34H
6) Асемблер:	ORL (direct), #<data>	
Код:	01000011	direct adress #data8
Час:	2 цикли	
Приклад:		;(P1)=00H
	ORL P1, #0C4H	;(P1)=11000100B (C4H)

Рисунок 4.39 – Приклад команди ORL <байт призначення>, <байт джерела>

Допускається 6 комбінацій режимів адресації:

– якщо байтом призначення є акумулятор:

1) регістрова;

2) пряма;

3) непряма;

4) безпосередня;

– якщо байтом призначення є пряма адреса:

5) неявна (акумулятор);

6) безпосередня.

Примітка. Якщо команда використовується для роботи з портом, величина, використовувана в якості вихідних даних порту, зчитується з "защіпки" порту, а не з виводів МК.

4.12.8.34 Команда ORL C, <біт джерела>

Команда "логічне "АБО" для змінних–бітів" (рисунок 4.40) встановлює прапорець перенесення C, якщо булева величина дорівнює логічній "1", у протилежному випадку скидає прапорець C в "0". Косий дріб ("/") перед операндом на мові асемблера вказує на те, що в якості операнда використовується логічне заперечення значення біта, що адресується, але сам біт джерела не змінюється. Ця команда на інші прапорці не впливає.

1) Асемблер:	ORL C, <bit>	
Код:	01110010	bit adress
Час:	2 цикли	
Приклад:		;(C)=00H, (P1)=53H (01010011B)
	ORL C, P1.4	;(C)=01H, (P1)=53H (01010011B)
2) Асемблер:	ORL C, /<bit>	
Код:	10100000	bit adress
Час:	2 цикли	
Приклад:		;(C)=0, (ОЗП[25])=39H (00111001B)
	ORL C, /2A	;(C)=1, (ОЗП[25])=39H (00111001B)

Рисунок 4.40 – Приклад команди ORL C, <біт джерела>

4.12.8.35 Команда POP <direct>

Команда "читання зі стеку" (рисунок 4.41) зчитує вміст комірки, яка адресується за допомогою покажчика стеку, в комірку ОЗП, яка адресується прямо, при цьому покажчик стеку зменшується на одиницю. Ця команда не впливає на прапорці і часто використовується для читання зі стеку проміжних даних.

```
Асемблер:   POP <direct>
Код:        11010000          direct adress
Час:        2 цикли
Приклад:                      ;(SP)=32H, (DPH)=01H, (DPL)=ABH
                              ;(ОЗП[32])=12H, (ОЗП[31])=56H
                              ;(ОЗП[30])=20H
            POP DPH
            POP DPL           ;(SP)=30H, (DPH)=12H, (DPL)=56H
                              ;(ОЗП[32])=12H, (ОЗП[31])=56H
            POP SP            ;(SP)=20H, (ОЗП[30])=20H
```

Рисунок 4.41 – Приклад команди POP <direct>

4.12.8.36 Команда PUSH <direct>

Команда "запис у стек" (рисунок 4.42) збільшує покажчик стеку на одиницю і після цього вміст зазначеної змінної, що адресується прямо, копіюється в комірку внутрішнього ОЗП, що адресується за допомогою покажчика стеку. На прапорці ця команда не впливає і використовується для запису проміжних даних у стек.

```
Асемблер:   PUSH <direct>
Код:        11000000          direct adress
Час:        2 цикли
Приклад:                      ;(SP)=09H, (DPTR)=1279H
            PUSH DPL
            PUSH DPH          ;(SP)=0BH, (DPTR)=1279H
                              ;(ОЗП[0A])=79H, (ОЗП[OB])=12H
```

Рисунок 4.42 – Приклад команди PUSH <direct>

4.12.8.37 Команда RET

Команда "повернення з підпрограми" (рисунок 4.43) послідовно вивантажує старший і молодший байти лічильника команд зі стеку, зменшуючи покажчик стеку на 2. Виконання основної програми зазвичай продовжується за адресою команди, наступної за ACALL або LCALL. На прапорці ця команда не впливає.

```
Асемблер:   RET
Код:        00100010
Час:        2 цикли
Приклад:                    ; (SP)=0DH,
                            ;(ОЗП[0C]=93H, (ОЗП[0D])=02H
            RET             ;(SP)=0BH, (PC)=0293H
```

Рисунок 4.43 – Приклад команди RET

4.12.8.38 Команда RETI

Команда "повернення з переривання" (рисунок 4.44) вивантажує старший і молодший байти лічильника команд зі стеку і встановлює "логіку обробки переривань", дозволяючи прийом інших переривань з рівнем пріоритету, який дорівнює рівню пріоритету щойно відпрацьованого переривання.

```
Асемблер:   RETI
Код:        00110010
Час:        2 цикли
Приклад:                    ;(SP)=0BH,
                            ;(ОЗП[0A])=2AH, (ОЗП[0B]=03H
                            ;(PC)=УУУУН, де У=0–FH
            RETI            ;(SP)=09H, (PC)=032AH
```

Рисунок 4.44 – Приклад команди RETI

Покажчик стеку зменшується на 2. Слово стану програми (PSW) не відновлюється автоматично. Виконання основної програми продовжується з команди, наступної за командою, на якій відбувся перехід за виявленим запитом на переривання. Якщо при виконанні команди RETI виявлено

переривання з таким самим або меншим рівнем пріоритету, то одна команда основної програми встигає виконатися до обробки такого переривання.

4.12.8.39 Команда RL A

Команда "зсув вмісту акумулятора вліво" (рисунок 4.45) зсуває 8 біт акумулятора на 1 біт вліво, біт 7 засилається на місце біта 0. На прапорці ця команда не впливає.

```
Асемблер:   RL A
Код:        00100011
Час:        1 цикл
Приклад:                    ;(A)=05H (11010101B), (C)=0
            RL A            ;(A)=0ABH (10101011B), (C)=0
```

Рисунок 4.45 – Приклад команди RL A

4.12.8.40 Команда RLC A

Команда "зсув вмісту акумулятора вліво через прапорець перенесення" (рисунок 4.46) зсуває 8 біт акумулятора і прапорець перенесення на 1 біт вліво.

```
Асемблер:   RLC A
Код:        00110011
Час:        1 цикл
Приклад:                    ;(A)=56H (01010110B), (C)=1
            RLC A           ;(A)=0ADH (10101101B), (C)=0
```

Рисунок 4.46 – Приклад команди RLC A

Вміст прапорця перенесення поміщається на місце біта 0 акумулятора, а вміст біта 7 акумулятора переписується в прапорець перенесення. На інші прапорці ця команда не впливає.

4.12.8.41 Команда RR A

Команда "зсув вмісту акумулятора вправо" (рисунок 4.47) зсуває вправо на 1 біт всі 8 біт акумулятора. Вміст біта 0 поміщається на місце біта 7. На прапорці ця команда не впливає.

Асемблер	RR A	
Код:	00000011	
Час:	1 цикл	
Приклад:		;(A)=0D6h (11010110), (C)=1
	RR A	;(A)=6Bh (01101011), (C)=1

Рисунок 4.47 – Приклад команди RR A

4.12.8.42 Команда RRC A

Команда "зсув вмісту акумулятора вправо через прапорець перенесення" (рисунок 4.48) зсуває 8 біт акумулятора вправо через прапорець перенесення. Біт 0 переміщається в прапорець перенесення, а початковий вміст прапорця перенесення поміщається в біт 7. На інші прапорці ця команда не впливає.

Асемблер	RRC A	
Код:	00010011	
Час:	1 цикл	
Приклад:		;(A)=0D6h (11010110), (C)=1
	RR A	;(A)=6Bh (11101011), (C)=0

Рисунок 4.48 – Приклад команди RRC A

4.12.8.43 Команда SETB <біт >

Команда "встановити біт" встановлює вказаний біт в 1(рисунок 4.49).

Адресується:

1) прапорець перенесення (C)–неявно;

2) біт–прямо.

1) Асемблер	SETB C	
Код:	11010011	
Час:	1 цикл	
Приклад:		;(C)=0
	SETB C	;(C)=1
2) Асемблер	SETB (bit)	
Код:	11010010	bit adress
Час:	1 цикл	
Приклад:		;(P2)=38h (00111000)
	SETB P2.0	
	SETB P2.7	;(P2) = B9h (10111001)

Рисунок 4.49 – Приклад команди SETB

4.12.8.44 Команда SJMP <мітка>

Команда "короткий перехід" (рисунок 4.50) виконує безумовне розгалуження в програмі за вказаною адресою. Адреса розгалуження обчислюється підсумовуванням зміщення зі знаком у другому байті команди з вмістом лічильника команд після додавання до нього 2. Таким чином, адреса переходу може перебувати в діапазоні від 128 байт, які передують наступній команді, до 127 байт після нею.

Асемблер	SJMP <мітка>	
Код:	10000000 <REL8>	
Час:	2 цикли	
Приклад:		;(PC) = 0418h, MIT1 = 039Ah
	SJMP MIT1	;(PC)=039Ah, (REL8) = 80h

Рисунок 4.50 – Приклад команди SJMP <мітка>

4.12.8.45 Команда SUBB A, <байт джерело>

Команда "віднімання з заємом" (рисунок 4.51) віднімає зазначену змінну із прапорцем перенесення від вмісту акумулятора, засилаючи результат в акумулятор. Ця команда встановлює прапорець перенесення (позики), якщо при відніманні для біта 7 необхідна позика, в протилежному випадку прапорець перенесення скидається. Якщо прапорець перенесення встановлено, то позика необхідна при відніманні зі

збільшеною точністю на попередньому кроці, тому прапорець перенесення віднімається від вмісту акумулятора разом з операндом джерела. Прапорець AC встановлюється, якщо позика необхідна для біта 3 і скидається в протилежному випадку. Прапорець переповнення (OV) встановлюється, якщо позика необхідна для біта 6, але її немає для біта 7; або є для біта 7, але немає для біта 6.

1) Асемблер	SUBB A, Rn,	де n=0..7
Код:	10011rrr,	де rrr=000..111
Час:	1цикл	
Приклад:		;(A) = 0C9h, (R2) = 54h, (C) = 1
	SUBB A, 2	;(A) = 74h, (R2) = 54h, (C) = 0,
		;(AC) = 0, (OV) = 1
2) Асемблер	SUBB A, <direct>	
Код:	10010101	direct adress
Час:	1цикл	
Приклад:		;(A) = 97h, (B) = 25h, (C) = 0
	SUBB A, B	;(A) = 72h, (B) = 25h, (C) = 0,
		;(AC) = 0, (OV) = 1
3) Асемблер	SUBB A, @Ri,	де i = 0,1
Код:	1001011i	
Час:	1 цикл	
Приклад:		;(A) = 49h, (33h) = 25h, (C) = 1
		;(RAM[33h]) = 68h
	SUBB A, @R0	;(A) = E0h, (C) = 1
		;(AC) = 0, (OV) = 0
4) Асемблер	SUBB A, #data	
Код:	10010100	#data8
Час:	1 цикл	
Приклад:		;(A) = 0BEh, (C) = 0
	SUBB A, #3fh	;(A) = 7fh, (C) = 0
		;(AC) = 1, (OV) = 1

Рисунок 4.51 – Приклад команди SUBB A, <байт джерело>

При відніманні цілих чисел зі знаком (OV) вказує на від'ємне число, яке отримується при відніманні від’ємної величини від додатної, або додатне число, яке отримується при відніманні додатного числа від від’ємного.

Операнд джерела допускає 4 режими адресації:

регістрова, пряма, непряма та безпосередня (константа).

4.12.8.46 Команда SWAP A

Команда "обмін тетрадами всередині акумулятора" (рисунок 4.52) здійснює обмін між молодшими і старшими чотирма бітами акумулятора (між старшою та молодшою тетрадами).

1) Асемблер	SWAP A	
Код:	11000100	
Час:	1 цикл	
Приклад:		;(A) = 0D7h, (11010111b)
	SWAP A	;(A) = 7Dh, (01111101b)

Рисунок 4.52 – Приклад команди SWAP A

Ця команда може розглядатися так само, як команда чотирьохбітового циклічного зсуву. На прапорці ця команда не впливає.

4.12.8.47 Команда XCH A, <byte>

Команда "обмін вмісту акумулятора зі змінною–байтом" (рисунок 4.53) виконує обмін вмісту акумулятора з вмістом джерела, вказаного в команді.

1) Асемблер	XCH A, Rn, де	n = 0–7
Код:	11001rrr, де	rrr=000–111
Час:	1цикл	
Приклад:		;(A) = 03Ch, (R4) = 15h
	XCH A, R4	;(A) = 15h, (R4) = 3Ch,
2) Асемблер	XCH A, <direct>	
Код:	11000101	direct address
Час:	1 цикл	
Приклад:		;(A) = 0FEh, (P3) = 0DAh
	XCH A, P3	;(A) = 0DAh, (P3) = FEh
3) Асемблер	XCH A, @Ri	i = 0,1
Код:	1100011i	
Час:	1 цикл	
Приклад:		;(A) = 0BCh, (R1) = 39h,
		;RAM[39] = 44h
	XCH A, @R1	;RAM[39] = 0BCh
		;(A) = 44h

Рисунок 4.53 – Приклад команди XCH A, <byte>

Операнд джерела може використовувати наступні режими адресації: регістрова, пряма та непряма.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Дайте визначення символічним позначенням:
    a. Rn
    b. @Ri
    c. DPTR
    d. @A+DPTR
    e. rel
2) Охарактеризуйте роль акумулятора А у командах пересилок.
3) Перелічить доступні у наборі команд МК–51 арифметичні операції.
4) В яких логічних операціях результат записується в акумулятор?
5) Роль бітового процесора в архітектурі МК–51?
6) Охарактеризуйте довгий та абсолютний переходи в пам’яті.
7) Поясніть поняття «мнемоніка команди» та «мнемокод».
8) Назвіть та опишіть формати команд.
9) Як розміщуються команди пам’яті програми?
10) Як впливають команди на прапорці?
11) Як розрахувати час виконання команд?.
12) Опишіть способи адресації операндів.

ЛІТЕРАТУРА [4–6, 13, 14]

ПРЕДМЕТНИЙ ПОКАЗЧИК

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Гилмор Ч. Введение в микропроцессорную технику/ Пер. С англ..– М.: Энероатомидат, 1984.

2. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропорцессорах. Учеб. Пособие для тех. Вузов. – М.: Высш. Шк., 1991.

3. Каспер Эрни Программирование на языке АССЕМБЛЕРА для микроконтроллеров семейства i8051. – М.: Горячая линия – телеком, 2003.

4. В.В. Сташин и др.. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990.

5. Боборыкин А.В. и др. Однокристальные микро–ЭВМ. – М.: “МИКАП”, 1994.

6. А.О.Новацький, П.М.Повідайко. Організація та застосування одно кристальної мікроЕОМ МК51. – Навчальний посібник. – Житомир, 2001.

7. Хвощ С.Т. и др. Микропроцессоры и микро–ЭВМ в системах автоматического управления. Справочник.– Л.: Машиностроение, 1987.

8. Вуд М.А. Микропроцессоры в вопросах и ответах. – М.: Энергоатомидат, 1990.

9. Алексенко А.Г. и др. проектирование радиоэлектронной аппаратуры. – М.: Радио и связь, 1984.

10. Гольденберг Л.М. и др. Цифровые устройства и микропроцессорные системы. Задачи и упражнения. Учеб. пособие для вузов. – М.: Радио и связь, 1992.

11. Методические указания к выполнению курсовых проектов по дисциплине “Проектирование и применение микропроцессорных систем” (задания на КП, требования к объему, указания к оформлению,

рекомендации по проектированию типовой гипотетической МПС контроля, управления и отображения). – К.: КПИ, 2005.

12. Мікроконтролер ADUC 847, data sheet.

13. C8051F045R–RUS.pdf, data sheet.

14. Гладштейн М. А. Микроконтроллеры смешанного сигнала C8051 Fxxx фирмы Silicon Laboratories. – М. : «Додэка», 2008.